Alexandre Keunecke Ignácio de Mendonça
e
Felipe Guimarães Carvalho

Desmontador e Depurador Redirecionáveis para ASIPs

Florianópolis – SC

Agosto / 2006

Alexandre Keunecke Ignácio de Mendonça
e
Felipe Guimarães Carvalho

Desmontador e Depurador Redirecionáveis para ASIPs

Projeto de Pesquisa para a elaboração do Trabalho de conclusão de Curso apresentado como exigência para a obtenção do título de bacharel em Ciências da Computação à Universidade Federal de Santa Catarina - UFSC, no curso de Ciências da Computação.

Orientador:
Prof. Dr. Luiz Cláudio Villar dos Santos

Banca:
Olinto José Varela Furtado
Luiz Fernando Friedrich

Departamento de Informática e Estatística - INE
Centro Tecnológico - CTC
Universidade Federal de Santa Catarina

Florianópolis – SC

Agosto / 2006

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação

---

Prof. Dr. Luiz Cláudio Villar dos Santos
Departamento de Informática e Estatística - INE
Orientador

Prof. Dr. Olinto José Varela Furtado
Departamento de Informática e Estatística - INE
Banca

Prof. Dr. Luiz Fernando Friedrich
Departamento de Informática e Estatística - INE
Banca

Resumo

Atualmente, projetistas de sistemas dedicados enfrentam novos desafios para a concepção de tais sistemas, principalmete após a difusão da tecnologia conhecida como SoC (System on a chip) que são sistemas integrados de software e hardware (CPU, memória, periféricos, etc). Tais sistemas requerem o chamado design space exploration, que consiste em verificar a melhor configuração de hardware/software para o sistema, o que pode ser demorado, para agilizar este processo ferramentas automaticamente redirecionáveis de geração e depuração de código são mandatórias.

Este trabalho descreve uma técnica para gerar automáticamente ferramentas de desmontagem e depuração para a inspeção de código, a técnica proposta se baseia em duas idéias básicas: extrair automáticamente informações dependentes de arquitetura de um modelo escrito numa ADL (Architecture Description Language), gerando módulos dependentes de arquitetura de um pacote de utilidades binárias. Utilizamos a ferramenta acdsm para gerar desmontadores a partir de modelos das arquiteturas MIPS, SPARC-V8, POWERPC, i8051 e o acgdb para gerar depuradores para as arquiteturas MIPS, SPARC-V8, e POWERPC. Evidências experimentais de funcionalidade são demonstradas neste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Os Sistemas Embutidos estão ficando cada vez mais populares, e tem sido usados em diversas aplicações como por exemplo:

- Eletrônica Automotiva: os carros modernos contém um significativo número de partes eletrônicas, como o controle dos freios ABS, sistema de rastreamento, entre outros.
- Telecomunicações: um exemplo disto é a telefônia móvel, mercado que teve um grande crescimento nos últimos anos.
- Eletrônica de Consumo: equipamentos de áudio e vídeo vem se tornando cada vez mais populares e mais sofisticados; portanto, necessitam de um melhor processamento de informação, necessitando de processadores.

> The size of embedded system market can be analyzed from a variety of perspectives. Looking at the number of processors that are currently used, it has been estimated that about 79% of all the processors are used in embedded systems.(MARWEDEL, 2003) [1]

Sistemas embutidos tem características especiais para medir eficiências em relação aos dois outros grandes mercados de processadores (Desktop e Servidores). Dentre tais características, destacam-se:

1. Energia: como grande parte dos Sistemas Embutidos rodam em baterias (celulares, palm, etc), eles devem ser eficientes no consumo de energia.
2. Tamanho do Código: a hierarquia de memória em um sistema embutido é limitada, e tipicamente não temos discos rígidos, portanto, o tamanho de código gerado deve ser o menor possível para a aplicação.

[1] O tamanho do mercado de sistemas embutidos pode ser analizado por variadas perspectivas. Olhando o número de processadores que estão atualmente em uso, é estimado que cerca de 79% de todos os processadores são usados em sistemas embutidos (Tradução Nossa).

1.1 Tema

1.1.1 Delimitação do Tema

Ao se desenvolver uma ferramenta de manipulação de código de máquina tem-se a necessidade de escrever um decodificador para poder manipulá-lo. Como atualmente há uma infinidade de arquiteturas de processadores e também existem diferentes sistemas operacionais, nós teríamos que desenvolver um desmontador diferente para cada combinação de arquitetura com sistema operacional. Com um desmontador redirecionável, se entraria com a descrição da arquitetura e o sistema operacional e o desmontador seria gerado automaticamente.

Como em sistemas embutidos o dito "Time to Market"é muito curto, ou seja, os produtos devem ser desenvolvidos rapidamente, necessitamos de uma ferramenta para localizarmos os erros de software e corrigí-los o mais rápidamente possível. Como foi dito anteriormente, existe uma infinidade de arquiteturas de processadores, e para cada uma delas necessitamos de um depurador diferente, com o depurador redirecionável, esta tarefa será automatizada, poupando bastante tempo para o desenvolvedor.

1.2 Objetivos

1.2.1 Gerais

Utilizar uma liguagem ADL (Architecture Descripion Language, em português Linguagem de Descrição de Arquiteturas), para descrever o conjunto de instruções do processador alvo, sendo possível assim a geração automática de desmontadores e depuradores com as características da arquitetura descritas no arquivo de entrada.

1.2.2 Específicos

- Identificação das funções de um depurador.
- Geração automática de desmontadores.
- Geração automática de depuradores.

1.3 Motivação

Um desmontador pode ser útil para muitas tarefas. Você pode ter um programa que necessita ser modificado mas só tem em mãos o código objeto. Com um desmontador, você pode converter este objeto em um arquivo de texto, o modificá-lo e submetê-lo a ferramenta que faz o papel contrário que é o montador, e gerar novamente o código objeto. O desmontador pode não fazer todo o trabalho, mas um bom desmontador torna o trabalho muito mais fácil.

Um depurador por sua vez tem um papel talvez até mais importante que o desmontador, quando as coisas não vão bem (o que é comum), é necessário entender como a execução de um programa está sendo efetivamente executada, para poder corrigir erros existentes, economizando várias horas de trabalho.

Com a infinidade de arquiteturas de processadores existentes, e com o crescimento da complexidade dos sistemas desenvolvidos, projetistas de sistemas embutidos vem encontrando cada vez mais dificuldades na concepção de tais sistemas, há cada vez mais a necessidade de ferramentas que auxiliem este processo, para torná-lo menos "doloroso". Desmontadores e depuradores são úteis para uma possível localização/correção de erros, podendo assim ajudar todo um grupo de desenvolvedores que trabalham com a geração de programas para diversas arquiteturas, tornando o seu trabalho cada vez mais rápido e eficiente.

1.4 Ferramentas Redirecionáveis

Ferramentas redirecionáveis extraem automaticamente as informações principais sobre uma CPU, a partir de sua descrição em alguma linguagem de descrição de arquitetura (ADL), gerando ferramentas executáveis para aquela CPU. A figura 1 ilustra o processo de exploração das alternativas de cpu utilizando o presente trabalho.

Dado um modelo de uma arquitetura descrito em uma liguagem ADL, este é submetido ao gerador das ferramentas, esse por sua vez gera um conjunto de ferramentas, tais como: compilador, montador, linker, desmontador, depurador e simulador. Após a geração das ferramentas, o código das aplicações pode ser compilado, montado e linkado, gerando um arquivo executável. Em um passo seguinte o código executável pode ser executado no simulador, e verificando sua funcionalidade com o auxílio do desmontador e depurador, após verificado o correto funcionamento deste, pode ser verificado se os requerimentos do

sistema tais como: tamanho de código e performance são preenchidos. Se os requerimentos não forem aceitos, pode-se gerar as ferramentas para outra CPU-alvo, e repetirmos o processo.

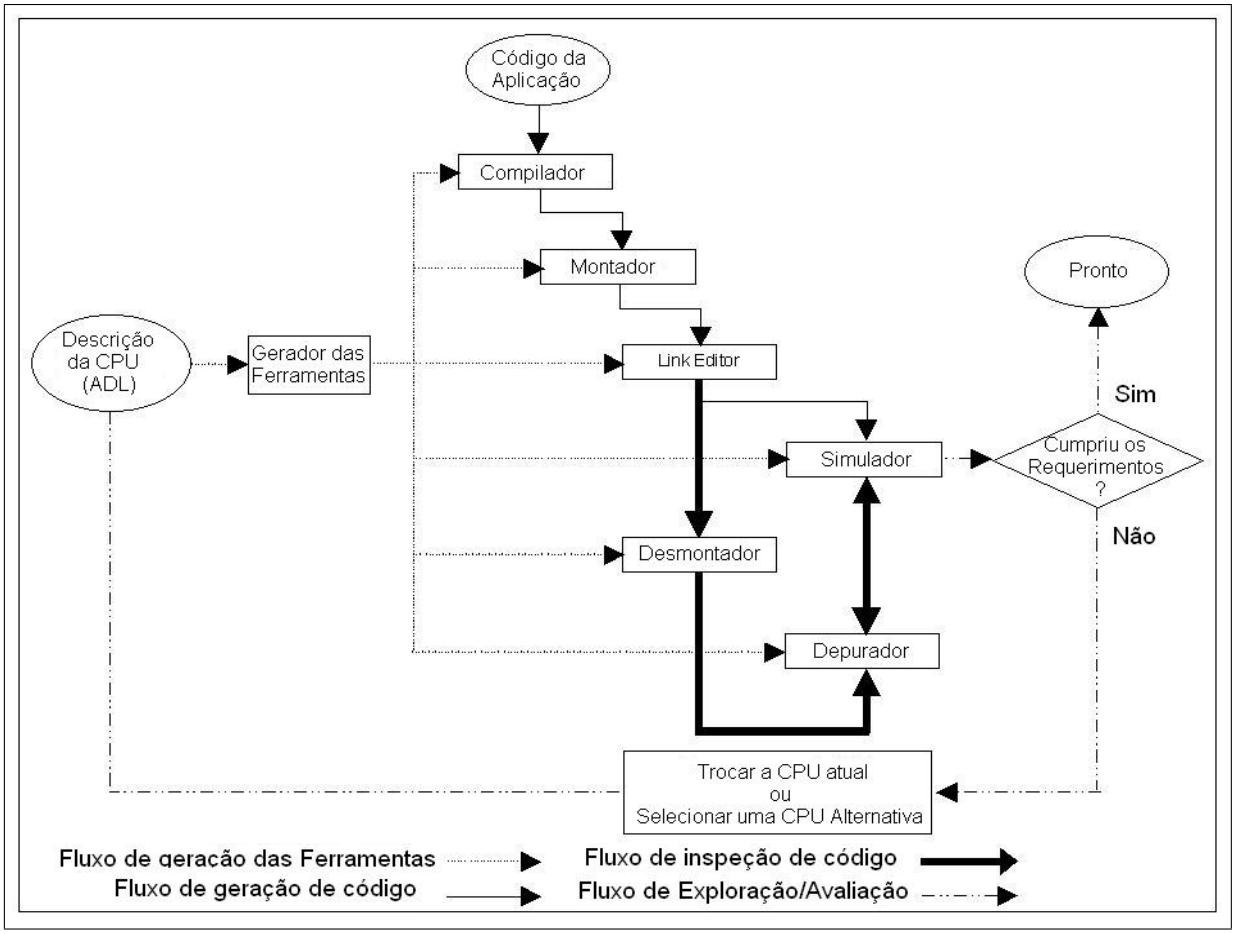


Figura 1: *Fluxo de exploração baseado em ADL*

1.5 Organização do Trabalho

O presente trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados alguns fundamentos sobre o pacote GNU Binutils e sobre a ADL ArchC, necessários para o desenvolvimento do trabalho. A geração automática de ferramentas redirecionáveis é abordada no Capítulo 3, onde descreve-se a técnica utilizada e a geração de arquivos necessários para o redirecionamento. No Capítulo 4, são descritas as técnicas usadas para a validação das ferramentas, assim como seus resultados experimentais. As conclusões são apresentadas no Capítulo 5.

1.6 A Contribuição Individual de um Trabalho Cooperativo

As contribuições técnicas deste trabalho inserem-se no contexto de um trabalho mais amplo, executado em cooperação e sob a supervisão de Max Ruben de Oliveira Schultz, aluno de mestrado do Programa de Pós-Graduação em Ciência da Computação da UFSC.

O aluno Max Schultz é o autor da proposta conceitual das ferramentas reconfiguráveis e da metodologia de validação experimental aqui relatada, enquanto a contribuição dos autores deste trabalho foi predominantemente técnica na implementação de uma ferramenta-protótipo e na execução dos experimentos sugeridos. Cabe-nos ressaltar que vários elementos do texto deste trabalho (figuras, descrições, tabelas, gráficos) são reproduções extraídas de artigos escritos em co-autoria com referido aluno e serão reutilizadas em sua futura dissertação de mestrado, uma vez que são produto de trabalho cooperativo.

2 Fundamentos

Neste capítulo serão abordadas as ferramentas utilizadas como base para o desenvolvimento deste trabalho. As duas pricipais seções, descrevem o "GNU Binutils", que trata superficialmente como as bibliotecas deste pacote são divididas e sua importância para a geração das ferramentas redirecionáveis, e a "Linguagem de descrição de Arquiteturas ArchC", que faz um histórico desta linguagem, logo após mostraremos um pequeno tutorial de como se escrevem modelos nesta linguagem para a geração automática das ferramentas redirecionáveis.

2.1 GNU Binutils

Binutils é um conjunto de ferramentas para a manipulação de arquivos objetos, como o seu nome sugere. O binutils faz parte da cadeia de ferramentas GNU para a construção de softwares, fazem parte desta cadeia também outros softwares como GNU Compiler Colection (gcc), GNU Assembler (gas), GNU Linker (ld), GNU Profiler (gprof), GNU Object Dump (objdump), entre outros.

A ferramenta GNU Debugger (gdb) fazia parte do pacote GNU Binutils, porém recentemente foi separada do pacote, seguindo seu próprio rumo. As bibliotecas do pacote GNU GDB são espelhos das do GNU Binutils, justificando assim a abordagem unificada das ferramentas.

Essencialmente, o pacote Binutils consiste de um módulo core indenpendente de arquitetura e alguns módulos dependentes de arquitetura que devem ser reescritos para cada nova CPU-alvo. Entre os módulos dependentes de arquitetura, existem duas principais bibliotecas, Opcodes e BFD, as quais requerem redirecionamento.

1. **BFD**

   BFD é um pacote que permite às aplicações usar as mesmas rotinas para operarem com diferentes formatos de arquivos objetos, e é dividida em duas partes:

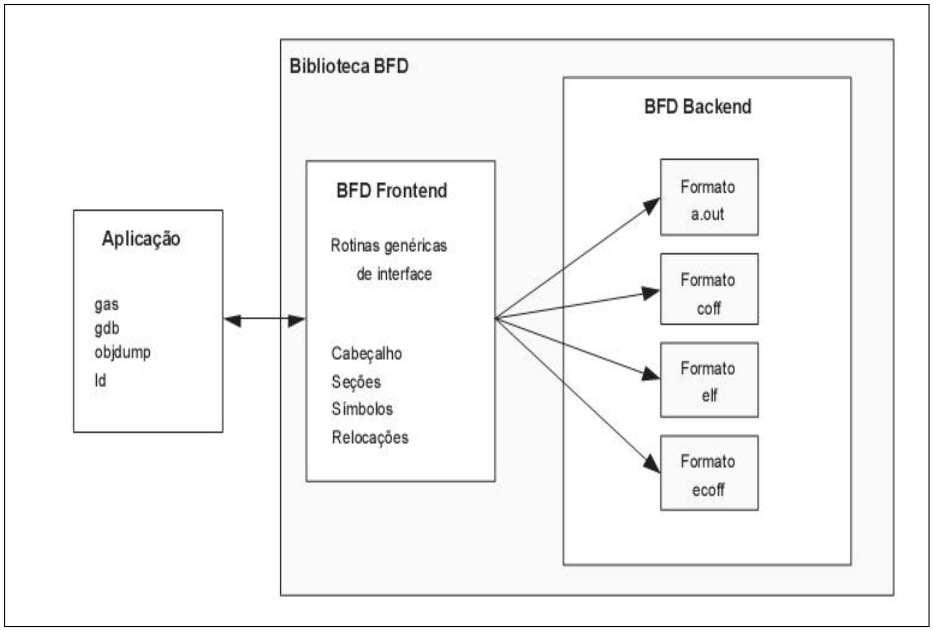


Figura 2: *Estrutura da biblioteca BFD*

- Frontend: A interface com o usuário, ele administra a memória e várias estruturas de dados canônicas. O Frontend também decide qual backend utilizar e quando chamar as rotinas do backend.
- Backend: Cada backend provê um conjunto de chamadas as quais o frontend do BFD pode usar para manter sua forma canônica. Os backends também podem manter informações para o seu próprio uso, para uma melhor eficiência. Um novo formato de arquivo objeto pode ser suportado simplesmente criando um novo BFD backend e adicionando-o a biblioteca.

  A figura 2 extraída de (BALDASSIN, 2005) mostra a estrutura da biblioteca BFD.

2. **Opcodes**

Esta biblioteca é responsável pela descrição da ISA (Instruction Set Architecture) de uma arquitetura e de informações sobre codificação e decodifição das instruções desta arquitetura. Nesta biblioteca se concentra o maior trabalho para se portar uma nova arquitetura no pacote. Ela é utilizada por outras ferramentas do pacote GNU Binutils como o objdump e o gdb, que são alvos deste trabalho e serão descritos a seguir.

Não existe uma estrutura comum de representação dos opcodes, fazendo assim com que cada arquitetura adote uma representação particular.

2.1.1 Objdump

O GNU Objdump é o desmotador do pacote GNU Binutils, o objdump é utilizado principalmente como ferramenta de verificação de arquivos objetos, e auxilia os desenvolvedores na detecção de erros. Ele permite o redirecionamento das arquiteturas utilizando as bibliotecas BFD e Opcodes, necessitando apenas que um conjunto de rotinas dependentes de arquitetura seja reescrito para cada nova arquitetura.

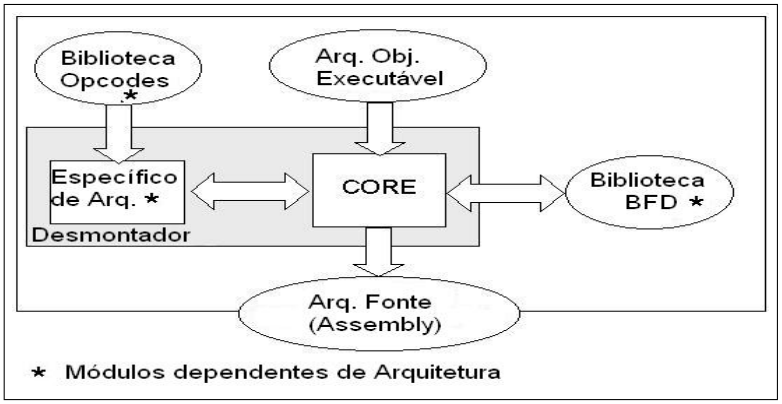


Figura 3: *Objdump e suas bibliotecas*

As ferramentas do GNU Binutils já foram redirecionadas para diversas arquiteturas CISC e RISC.

2.1.2 GNU Debugger (gdb)

O GDB (Gnu Debugger) é o depurador do pacote Binutils, basicamente o seu redirecionamento é composto de duas partes: a primeira é semelhante a do objdump (utilizando as bibliotecas BFD e Opcodes), a segunda necessita que um módulo do depurador seja reescrito para cada nova arquitetura, este módulo é responsável pelos breakpoints, chamadas de procedimentos, etc.

2.2 Linguagem de descrição de Arquiteturas ArchC

ArchC é uma linguagem de descrição de arquiteturas (ADL) pública. ArchC é a primeira ADL capaz de gerar simuladores escritos em SystemC. ArchC foi criada para que projetistas de arquitetura pudessem contar com uma ferramenta que lhes permitissem avaliar rapidamente o efeito de novas idéias na área de arquitetura de computadores, tendo em vista esse objetivo, os focos inicias foram a geração automática das ferramentas de

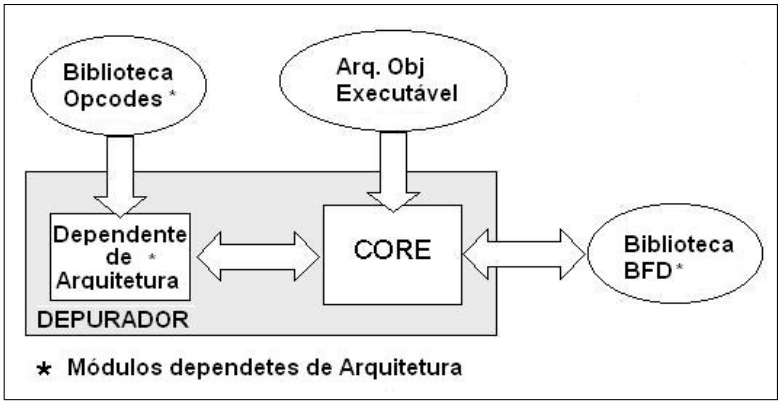


Figura 4: *GDB e suas bibliotecas*

simulação e co-verificação. Além disso, ArchC foi projetada para dar suporte a geração de outras ferramentas como montadores, ligadores, desmontadores, depuradores, etc.

2.2.1 Descrição da Arquitetura

Uma descrição de arquitetura em ArchC se faz em duas partes: uma na qual se descreve o conjunto de instruções (*AC_ISA*) e a outra na qual se descreve os recursos da arquitetura (*AC_ARCH*). Na descrição *AC_ISA* está a informação sobre os formatos das instruções, a declaração das instruções e suas propriedades e informações necessárias para a decodificação das instruções. Em um arquivo separado, escrito em linguagem C/C++, descreve-se o comportamento de cada instrução declarada. Na descrição *AC_ARCH* está toda a informação sobre a arquitetura necessária para a geração automática das ferramentas de software. As informações sobre estrutura de pipeline, dispositivos de armazenamento (registradores, banco de registradores, memória) e outras informações da estrutura da arquitetura podem ser descritos. A figura 5 extraída de (BALDASSIN, 2005) mostra a idéia em termos de blocos de descrição.

Vamos ilustrar na figura 6 uma descrição dos recursos da arquitetura. Uma descrição de recursos sempre começa com a palavra-chave *AC_ARCH* e deve ser também informado entre parênteses o nome do projeto (ex.: mips1). Na linha 3 a construção *ac_wordsize* define o tamanho da palavra da arquitetura. A definição de um dispositivo de armazenamento é feito na linha 5 pela construção *ac_cache* seguido do nome do dispositivo, dois pontos e o tamanho da memória. Na linha 6 define-se um dispositivo de armazenamento do tipo "banco de registradores"através da construção *ac_regbank* seguido do nome do objeto, dois pontos e a quantidade de registradores disponíveis. A construção *ARCH_CTOR* inicia a declaração do construtor, como mostrado na linha 8. Na linha 10 a construção

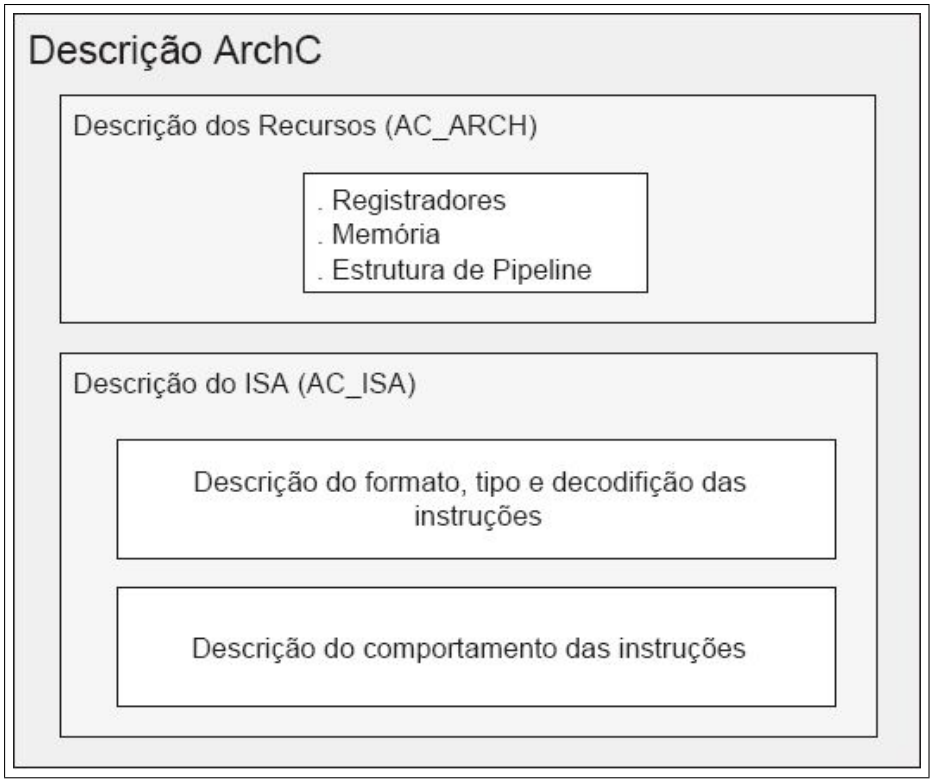
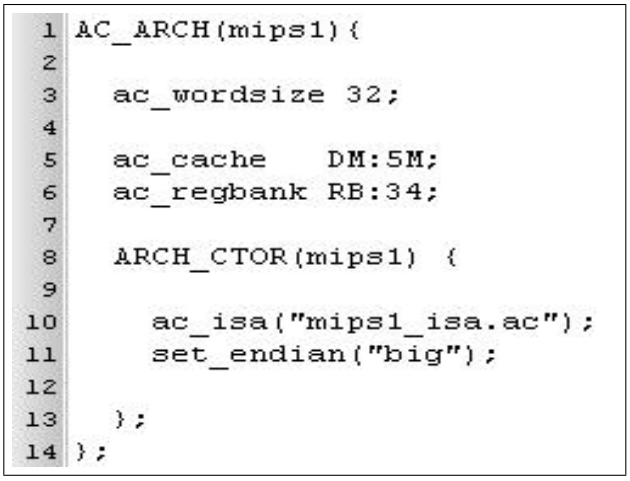


Figura 5: *Estrutura de Descrição em ArchC*

ac_isa informa o nome do arquivo que contém a descrição das instruções (*AC_ISA*). A configuração do endianness da arquitetura para big ou little é feita na linha 11 pela construção *set_endian* . O modelo descrito na figura 3.3, é um modelo funcional, ou seja, um modelo que é capaz de executar todas as instruções de uma arquitetura.

A figura 7 ilustra um exemplo da descrição do conjunto de instruções. Uma descrição de instruções sempre começa com a palavra-chave *AC_ISA* e deve ser também informado entre parênteses o nome do projeto (ex.: *mips1*), como na descrição *AC_ARCH* . Nas linhas 3 a 5 a construção *ac_format* define os formatos possíveis das instruções (ex.: Type_R , Type_I , Type_J), declarando seus campos (ex.: op, rs, rt, rd, imm , etc.) e o tamanho de cada campo. Nas linhas 7 a 9 a construção *ac_instr* associa cada instrução a um tipo pré-definido pelo formato de instrução (toda a instrução em ArchC deve estar associada a um formato). O mapeamento entre nomes simbólicos e registradores da CPU é definido nas linhas 11 a 20 pela construção *ac_asm_map_reg* . Como cada instrução possui suas próprias propriedades, então foram criados métodos para poder-se configura-los. Nas linhas 24, 27 e 30 é definido a sintaxe assembly das instruções pela construção *set_asm* . A construção *set_decoder* nas linhas 25, 28 e 31 atribui valores aos campos do formato amarrado a instrução, que serão usados para decodificação de uma instrução.



```
1  AC_ARCH(mips1){
2
3    ac_wordsize 32;
4
5    ac_cache   DM:5M;
6    ac_regbank RB:34;
7
8    ARCH_CTOR(mips1) {
9
10     ac_isa("mips1_isa.ac");
11     set_endian("big");
12
13   };
14 };
```

Figura 6: *Descrição da declaração de recursos de um modelo funcional do MIPS escrito em ArchC*

Na linha 33 é definida uma pseudo-instrução pela construção *pseudo_instr* , onde indica que a instrução (não associada a nenhuma instrução declarada através de *ac_instr*), deve executar a lista a instruções que descrevem a pseudo-instrução.

ArchC possui um pré-processador (acpp) que é composto por analisadores léxico, sintático e semântico que foram gerados pelas ferramentas GNU flex e GNU bison. O pré-processador é responsável pela expansão de macros, substituição de pseudo-instruções por instruções nativas, cálculo de endereços-alvo de desvio, etc. O pré-processador obtém as informações necessárias da descrição de conjunto de instruções (*AC_ISA*) e da descrição de recursos da arquitetura (*AC_ARCH*) para a geração de classes C++ e/ou módulos SystemC que farão parte do código das ferramentas (simuladores, montadores, etc.). Importante ressaltar que o pré-processador não é utilizado diretamente pelos usuários e sim pelas ferramentas.

```
1  AC_ISA(mips1){
2
3    ac_format Type_R  = "%op:6 %rs:5 %rt:5 %rd:5 %shamt:5 %func:6";
4    ac_format Type_I  = "%op:6 %rs:5 %rt:5 %imm:16:s";
5    ac_format Type_J  = "%op:6 %addr:26";
6
7    ac_instr<Type_I> sb, sh, sw, swl, swr;
8    ac_instr<Type_R> add, addu, sub, subu, slt, sltu;
9    ac_instr<Type_J> j, jal;
10
11   ac_asm_map reg {
12      "$"[0..31] = [0..31];
13      "$zero" = 0;
14      "$at" = 1;
15      "$kt"[0..1] = [26..27];
16      "$gp" = 28;
17      "$sp" = 29;
18      "$fp" = 30;
19      "$ra" = 31;
20   }
21
22   ISA_CTOR(mips1){
23
24     sb.set_asm("sb %reg, %imm(%reg)", rt, imm, rs);
25     sb.set_decoder(op=0x28);
26
27     add.set_asm("add %reg, %reg, %reg", rd, rs, rt);
28     add.set_decoder(op=0x00, func=0x20);
29
30     j.set_asm("j %expA", addr);
31     j.set_decoder(op=0x02);
32
33     pseudo_instr("subu %reg, %reg, %imm") {
34       "addiu %0, %1, -%2";
35     }
36   };
37
38 };
```

Figura 7: *Trecho de uma descrição ISA para o MIPS escrito em ArchC*

3 As Ferramentas Redirecionáveis acdsm e acgdb

Neste capítulo trataremos da geração das ferramentas redirecionáveis, bem como da técnica adotada para a conversão dos dados do modelo ArchC para as estruturas dependentes de arquitetura do binutils.

A técnica de geração automática adotada utiliza os pacotes GNU Binutils e GNU GDB como base, como visto na figura 8, note que o desmontador faz uso das bibliotecas Opcodes e BFD, e o depurador faz uso da biblioteca BFD, porém o depurador faz uso do desmontador para exibir as instruções em linguagem de montagem.

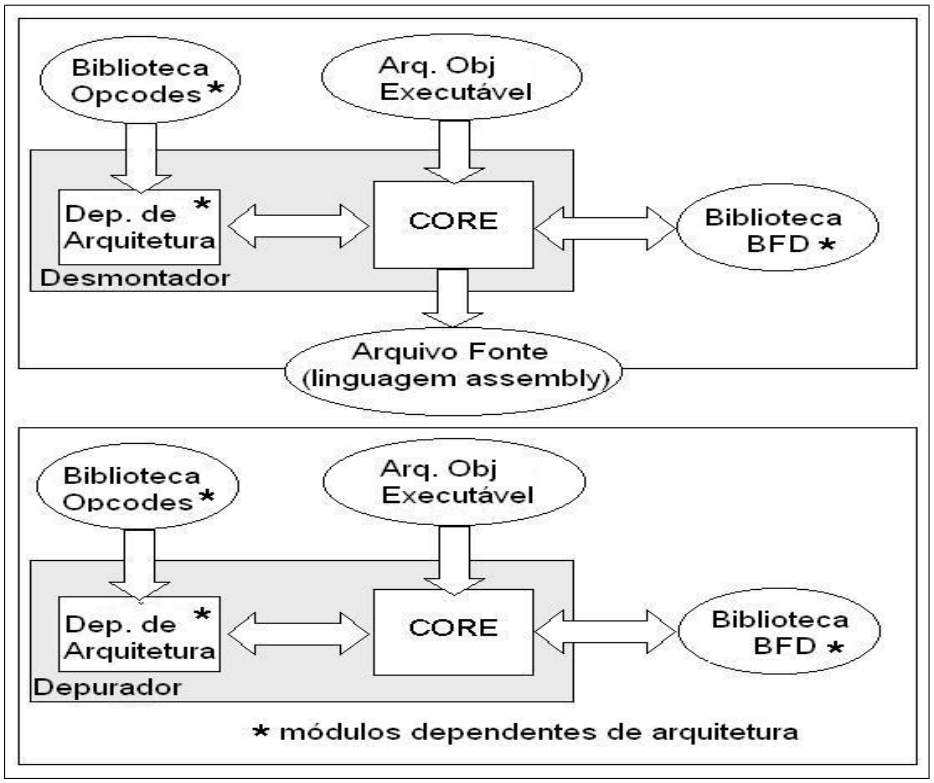


Figura 8: *Estrutura das Ferramentas*

A chave para a geração automática está em utilizar o máximo possível o Core das

ferramentas, que são independentes de arquitetura, e gerar automaticamente as bibliotecas BFD e Opcodes(dependentes de arquitetura) e os módulos dependetes de arquitetura de cada ferramenta atráves do modelo ArchC. Assim, compilando esses módulos com o core do pacote, gera-se a ferramenta executável.

Cada biblioteca utilizada consiste de alguns arquivos dependentes de arquitetura e outros que declaram as arquiteturas disponíveis ao core e qual rotina invocar para cada ação, dada uma arquitetura-alvo a qual chamaremos de [arq], a organização dos arquivos utilizados pelas ferramentas estão descritas na figura 9: os '-' denotam diretórios e os '.' denotam arquivos. O pacote GDB fazia parte do pacote binutils, e agora toma um rumo separado, por isso as bibliotecas opcodes e BFD são repetidas para o binutils e GDB como pode ser visto na figura 9.

```
-binutils/
   -opcodes/
      .disassemble.c
      .[arq]-opc.c
      .[arq]-dis.c
   -bfd/
      .cpu-[arq].c
      .elf32-[arq].c
   -include/
      -elf/
         .[arq].h
      -opcode/
         .[arq].h
-gdb/
   -opcodes/
      .disassemble.c
      .[arq]-opc.c
      .[arq]-dis.c
   -bfd/
      .cpu-[arq].c
      .elf32-[arq].c
   -include/
      -elf/
         .[arq].h
      -opcode/
         .[arq].h
   -gdb/
      .[arq]-tdep.c
      -config
         -[arq]
            .[arq].mt
```

Figura 9: *Arquivos afetados no redirecionamento das ferramentas*

3.1 Geração da Biblioteca Opcodes

A biblioteca Opcodes é responsável pela ISA de uma arquitetura e de informações sobre codificação e decodificação das intruções desta arquitetura. Essa biblioteca é escrita em C e além de ser usada pelo Objdump e GDB também é utilizada pelo montador do pacote Binutils (GAS).

Como já foi dito anteriormente, não existe uma estrutura padrão para a descrição das instruções. O Objdump faz uso desta biblioteca para a decodificação das instruções desta arquitetura e para saber o formato como o código assembly está organizado, como: estilo dos mnemônicos, nomes dos registradores, separação entre os operandos, etc. O GDB faz uso dessa biblioteca através do Objdump, quando precisa exibir uma instrução assembly para o usuário.

O arquivo *opcodes/disassemble.c* tem a função de declarar todas as arquiteturas existentes no pacote, bem como indicar ao core qual rotina de desmontagem é apropriada para cada arquitetura, para cada nova arquiteura gerada no pacote, devemos declará-la e indicar qual o método (localizado no arquivo *opcodes/[arq]-dis.c*) de desmontagem apropriado para esta arquitetura.

O arquivo *include/opcodes/[arq].h* declara 3 estruturas de dados, as quais descrevem o banco de registradores, a sintaxe das instruções assembly, a imagem binária básica, a máscara para desmontagem, pseudo-instruções, entre outros. Estas estruturas serão povoadas no arquivo *opcodes/[arq]-opc.c* .

A biblioteca Opcodes é gerada automaticamente através do modelo da arquitetura da seguinte maneira:

- A tabela dos registradores é extraída através da construção *ac_asm_map* descrita no modelo, povoando assim a estrutura *acasm_symbol* no arquivo *opcodes/[arq]-opc.c* estrutura essa que foi declarada no arquivo *include/opcodes/[arq].c* e é representada através de tuplas (símbolo, formatador, valor).

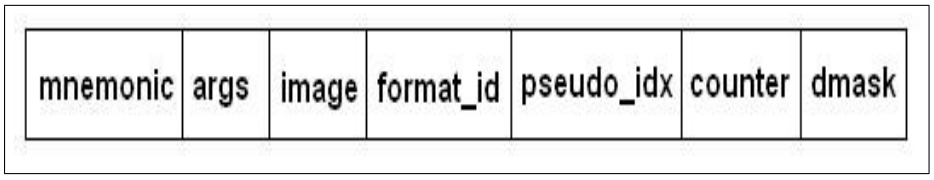


Figura 10: *Estrutura de Opcodes para uma instrução*

- A tabela de opcodes da arquitetura é separada em 7 partes, como visto na figura 10, a seguir iremos descrever cada uma delas:

  mnemonic: Mnemônico da instrução, e é separado através do primeiro espaço em branco entre o nome da instrução e seus operandos em *set_asm()* .

  args: argumentos da instrução, seguem o seguinte formato: " *%¡fmt¿:¡fid¿:¡reloc¿:* ", onde *¡fmt¿* é o formatador definido pelo usuário no modelo, ou predefinido pela linguagem, *¡fid¿* é um índice para a posição a qual este operando se refere na instrução, e *¡reloc¿* é uma informação referente ao Linker, explicitando se a instrução deve ou não ser relocada.

  image: imagem binária base da instrução, esta informação é obtida através da construção *set_decoder()* do modelo da arquitetura, representa a parte fixa do código binário para aquela instrução. Pseudo-instruções possuem valor padrão 0x00 para este campo.

  format_id: Identificador do formato de instrução. Cada formato de instrução é definido através da construção *ac_format()* no modelo da arquitetura e tem um identificador associado pelo pré-processador ArchC. Este identificador é utilizado posteriormente para obter-se o tamanho de cada campo da instrução através do formato da instrução, para ser feita a decodificação.

  pseudo_idx: Contém um índice para a tabela de pseudo-instruções, lembrando que uma pseudo-instrução pode ser composta de uma ou mais instruções nativas. Este campo é utilizado somente pelo montador, já que o desmontador não trabalha com pseudo-instruções, pois pode gerar ambigüidade.

  counter: Usado para contar o número de instruções montadas pelo montador (Opcional).

  dmask: Máscara utilizada para a desmontagem das instruções, é gerada com base nos operandos fixos da instrução e em sua imagem binária básica (setada por *set_decoder()*), ou seja a máscara aponta ao desmontador juntamente com a imagem base quais campos são importantes para a identificação de que instrução se trata.

- A tabela de pseudo-instruções é extraída através do comando *pseudo_instr()* do modelo, essa tabela é somente utilizada pelo montador, pois para a desmontagem pode gerar ambiguidade, por isso optamos por utilizar somente instruções nativas na desmontagem, pois o arquivo objeto é composto de instruções nativas.

3.2 Geração da Biblioteca BFD

Alguns atributos da aquitetura alvo extraídos do modelo ADL são codificados nesta biblioteca. Como adotamos o formato de arquivo ELF, somente os arquivos relacionados com este formato serão gerados. Os arquivos mais importantes para esta biblioteca são:

- *bfd/elf32-[arq].c* Basicamente fornece ao core algumas informações como nome do formato, tamanho máximo de página, qual a função que irá tratar as relocações, entre outros.
- *bfd/cpu-[arq].c* Arquivo mais importante da BFD, contem informações como: bits em uma palavra, bits em um endereço, nome da arquitetura, entre outros.

3.3 Módulo Dependente de Arquitetura do Desmontador

Este módulo é o principal em se tratando do desmontador, ele utiliza as bibliotecas Opcodes e BFD intensamente para a desmontagem. A seguir iremos descrever em alguns passos como é descrito este arquivo, e os algoritmos utilizados para a identificação das instruções.

Os passos 1A e 1B são mutuamente exclusivos, se a arquitetura em questão tiver tamanho de palavra fixo (usualmente em processadores RISC) utilizamos o passo 1A, porém se a arquitetura tiver tamanho de palavra que varia de instrução para instrução (usualmente em máquinas CISC) então optamos pelo passo 1B, neste passo existem algumas limitações, o tamanho de cada instrução da arquitetura deve ser múltiplo de 8, no intervalo entre [8,32], esta limitação será resolvida em breve, adicionando algumas informações ao modelo ArchC para melhor generalizar o desmontador e melhorar ainda mais a performance.

Passos para a decodificação das instruções:

1A (Para arquiteturas com tamanho de palavra fixa)- Para cada instrução presente no arquivo objeto, o core chama a função *print_insn_[arq]* , essa função utiliza a biblioteca BFD para ler X bits do arquivo objeto, onde X é o tamanho da palavra da arquitetura, esses bits são salvos em uma variável, que contém após esse passo a instrução pura(a qual chamaremos de raw insn), ou seja os bits que compõe essa instrução, após isso chamamos a função *disassemble()* passando como parametro a raw insn.

1B (Para arquiteturas com tamanho de palavra variável)- Para cada instrução presente no arquivo objeto, o core chama a função *print_insn_[arq]* , Para cada tamanho de palavra no intervalo [8,32], é lido do arquivo objeto através da biblioteca BFD, o número de bits correspondente, após isso, a tabela de opcodes é varrida procurando se a instrução é canditada para a desmontagem utilizando a raw insn, a máscara de desmontagem e a imágem binária básica, se os testes forem positivos, salvamos seu raw insn, e partimos para o próximo tamanho de palavra até o tamanho máximo, se neste intervalo for encontrada outra insn que passe nos testes, a anterior é substituida por esta, assim ao final do processo, temos o raw insn com o tamanho certo, o qual passamos como parâmetro para a função *disassemble()* .

2 - Para verificarmos de qual instrução se trata esta raw insn, varremos a tabela de instruções da arquitetura (declarada no arquivo *opcodes/[arq]-opc.c*) e para cada instrução aplicamos a máscara(dmask) na raw insn e fazemos um teste, se o resultado da aplicação da máscara corresponder a imágem binária básica da instrução (campo image), e essa instrução em questão não for uma pseudo-instrução então, podemos afirmar que essa é a instrução que queremos.

Para efeitos de exemplo, vamos supor que a instrução localizada acima se trata de uma instrução de add imediato do MIPS, essa instrução tem a seguinte sintaxe: " *addi $reg $reg $imm* ", essa instrução é tratada do modelo, gerando a seguinte estrutura na tabela de opcodes: {"addi", "%reg:2:0:,%reg:1:0:,%exp:3:1:", 0x20000000, 1, 0, 0, 0xFC000000} lembrando os campos: mnemônico, argumentos, image, id do formato, pseudo, counter, dmask.

3 - Imprimimos o mnemônico da instrução (" *addi* "), e a seguir partimos a procura de registradores e endereços da instrução (argumentos), para isso é chamado o método parse() que nos retorna uma lista ligada com os elementos da instrução (retirados do campo args da tabela de opcodes) e suas posições na instrução. No exemplo seria retornado uma lista de 3 posições com as seguintes tuplas: (reg,2),(reg,1),(exp,3)

4 - Para cada tupla da lista retornada por *parse()* fazemos:

Com base no id do formato da instrução retirado da tabela de opcodes (no nosso exemplo "1") e a posição deste campo na instrução segundo argumento da tupla. É então chamamada uma função auxiliar que recebe como parâmetro o segundo elemento da tupla, este método retorna o tamanho do campo em bits. Com base neste tamanho e posição do elemento na instrução, formamos uma nova máscara que irá servir para extrair da raw insn o valor binário para este campo. Então varremos a tabela de registradores para ver

se existe algum registrador com o valor recém extraído pela máscara, se encontrarmos tal registrador, imprimimos o símbolo do mesmo. Se não for encontrado nenhum registrador com este valor, estamos tratando de um campo imediato (que pode ser um endereço, uma expressão,etc), então imprimimos este valor em hexadecimal.

Quando terminarmos de processar todas as tuplas da lista retornada por parse, podemos afirmar que terminaram os argumentos da instrução em questão, então retornamos o número de bytes processados e retomamos o processo para a próxima instrução (passo 1).

3.4 Módulo Dependente de Arquitetura do Depurador

Este módulo é o mais importante para o redirecionamento do GDB, ele trata principalmente do tratamento dos frames, breakpoints, e chamadas de procedimentos, os quais serão descritos a seguir.

O primeiro passo na execução do desmontador, o core chama a função *_initialize_[arq]_tdep()* que indica ao core, quais funções deste arquivo irão tratar alguns aspectos da arquitetura, a seguir iremos citar alguns desses métodos e qual sua funcionalidade dentro do depurador.

- *set_gdbarch_register_type()* : recebe como parâmetro o número do registrador no banco de registradores e tem a função de retornar qual o tipo do mesmo, exemplos: inteiro 32 bits com sinal, ponto flutuante 32 bits, e assim sucessivamente.

- *set_gdbarch_register_name()* : recebe como parâmetro o número do registrador no banco de registradores e tem a função de retornar qual o nome simbólico do registrador, este método varre a estrutura *acasm_symbol* da biblioteca opcodes procurando pelo registrador solicitado.

- *set_gdbarch_inner_than()* : indica ao core a direção do crescimento da pilha, pode receber dois tipos de dados: *core_addr_lessthan* indicando que a pilha cresce em direção a endereços menores ou *core_addr_greaterthan* que por sua vez indica que a pilha cresce em direção a endereços maiores.

- *set_gdbarch_breakpoint_from_pc()* : indica ao core qual função que faz o tratamento de breakpoints, genericamente um breakpoint é um local no programa designado pelo usuário, onde este quer assumir o controle de execução do programa sempre que a

execução do mesmo alcançar o ponto onde foi setado o breakpoint. O procedimento *set_gdbarch_breakpoint_from_pc()* tem a função de retornar qual o código opcode da instrução *break* da arquitetura em questão.

- *set_gdbarch_print_insn()* : indica ao core qual a função de desmontagem será utilizada para imprimir as instruções da arquitetura, essa função pertence ao módulo dependente de arquitetura do desmontador, abordado na seção 3.3 deste mesmo capítulo.

- *[arq]_frame_cache* : simplificadamente, este método tem a função de criar frames para o tratamento de subrotinas, ele analiza o prólogo da subrotina para produzir um backtrace e permitir ao usuário a manipulação de variaveis e argumentos de frames antigos, o gdb necessita encontrar o endereço base de frames antigos, e descobrir onde os registradores deste frame foram salvos. Também guarda o endereço de retorno no frame, salva o contexto (valores de registradores), entre outras funções.

4 *Validação e Resultados Experimentais*

Neste capítulo, demonstraremos as técnicas de validação das ferramentas **acdsm** e **acgdb** , bem como os resultados experimentais gerados pelo processo de validação.

4.1 Validação do desmontador

Para a validação da ferramenta foi adotado o já conhecido benchmark Mibench rodando em três diferentes CPUs (MIPS, PowerPC, SPARC). Como esse conjunto de benchmarks requer manipulação de arquivos, uma característica mal apropriada para microcontroladores, foi utilizado o conjunto mais simples de benchmarks, o Dalton para a CPU-alvo i8051. Para a geração de montadores foi utilizado a ferramenta acasm e para a geração de desmontadores, foi utilizado a nossa ferramenta acdsm.

4.1.1 Validação das Ferramenta Geradas

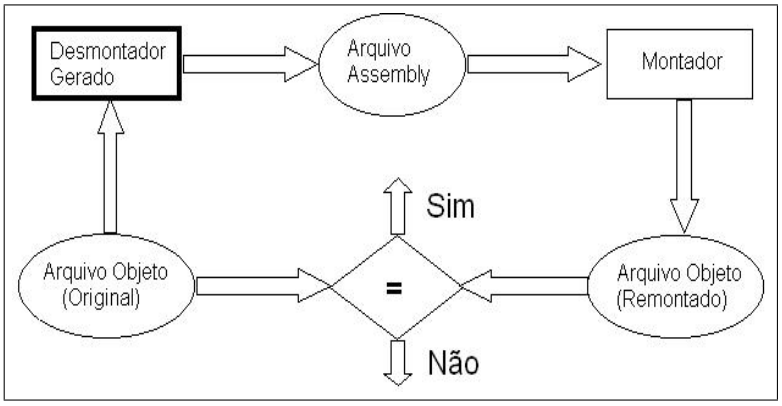


Figura 11: *Fluxo de validação dos desmontadores gerados*

Para validar as ferramentas geradas, foi seguido o seguinte procedimento, como visto na figura 11:

Passo 1 : Dado a descrição da CPU-alvo, acdsm gerou um desmontador, enquanto o acasm gerou um montador.

Passo 2 : Dado um arquivo objeto de entrada, este foi alimentado ao desmontador, gerando um arquivo de saída em linguagem assembly.

Passo 3 : Como o objdump é utilizado apenas para a verificação dos arquivos objetos, para realizarmos a remontagem do código assembly, necessitamos utilizar um pré-processador, que retira informações desnecessárias e adiciona labels para os desvios, após o pré-processamento, o arquivo assembly foi alimentado ao montador, tendo como saída um arquivo objeto.

Passo 4 : Os arquivos objetos montado e remontado foram comparados para verificar se as suas seções *".text"* se equivalem ou não. Foram comparadas somente as seções *".text"* dos arquivos, pois é nesta que se encontra o código executável da aplicação.

Como todas as comparações não diferiram para cada programa do benchmark e para cada CPU-alvo, existe uma forte evidência de validação correta.

Pode ser argumentado que tal procedimento deveria comparar os códigos assembly (entrada x saída). Porém, a comparação direta dos códigos assembly pode gerar um problema devido a presença de pseudo-instruções ou instruções com mais de uma sintaxe assembly. Por exemplo, a instrução "jump at register"do MIPS pode ser escrita de duas diferentes maneiras: *"jr $1"* ou *"j $1"*. Por isso foi utilizado a comparação dos arquivos objetos, assim eliminando situações ambiguas.

4.1.2 Validação da Ferramenta Geradora

Para verificar a redirecionabilidade da ferramenta geradora, o procedimento acima foi repetido para as CPU-alvo RISC (MIPS,SPARC,PowerPC) e CISC (i8051), os resultados são mostrados nas tabelas 1, 2, 3 e 4, respectivamente. A primeiras duas colunas mostram os nomes dos programas do benchmark e seus respectivos números de arquivos. As demais colunas mostram o tamanho da seção " *.text* "dos arquivos objetos em bytes, e os tempos de execução do desmontador (nativo e gerado) para cada CPU-alvo, como não existe um porte do binutils para o i8051 a coluna do tempo de execução da ferramenta nativa para esta CPU foi suprimida.

Pode ser observado através dos gráficos e tabelas, que os desmontadores gerados com

o acdsm tiveram uma performance um pouco inferior das obtidas com o desmontador nativo do GNU Binutils (objdump), fato justificado pela generalidade da ferramenta.

			Tempo de desmontagem (s)	
Nome	Num Arqs	Tam .text(bytes)	Acdsm	Nativo
basicmath_small	4	5148	0.007	0.007
basicmath_large	4	6336	0.012	0.009
bitcount	9	4992	0.010	0.006
qsort_small	1	1124	0.003	0.003
qsort_large	1	1896	0.003	0.004
susan	1	64720	0.099	0.074
jpeg	60	284868	0.442	0.317
typeset	1	29760	0.049	0.035
dijkstra_small	1	2564	0.004	0.006
dijkstra_large	1	2564	0.003	0.004
ispell	1	1232	0.002	0.003
stringsearch_small	4	5696	0.011	0.009
stringsearch_large	4	5696	0.012	0.008
blowfish	7	19208	0.036	0.028
sha	2	3324	0.007	0.005
rawcaudio	2	2592	0.006	0.005
rawdaudio	2	2536	0.002	0.003
crc32	1	1380	0.002	0.003
fft	3	5964	0.010	0.007

Tabela 1: *Resultados experimentais do acdsm para o MIPS*

			Tempo de desmontagem(s)	
Nome	Num Arqs	Tam .text(bytes)	Acdsm	Nativo
basicmath_small	4	5280	0.013	0.008
basicmath_large	4	6336	0.013	0.012
bitcount	9	4104	0.010	0.009
qsort_small	1	1184	0.004	0.005
qsort_large	1	2076	0.006	0.008
susan	1	59456	0.139	0.057
jpeg	60	239884	0.537	0.219
typeset	1	32664	0.071	0.031
dijkstra_small	1	2456	0.005	0.008
dijkstra_large	1	2456	0.006	0.006
ispell	1	1184	0.004	0.006
stringsearch_small	4	5340	0.012	0.010
stringsearch_large	4	5608	0.014	0.011
blowfish	7	16496	0.041	0.020
sha	2	2856	0.006	0.007
rawcaudio	2	2400	0.006	0.006
rawdaudio	2	2376	0.004	0.006
crc32	1	1268	0.001	0.006
fft	3	5564	0.014	0.010

Tabela 2: *Resultados experimentais do acdsm para o SPARC*

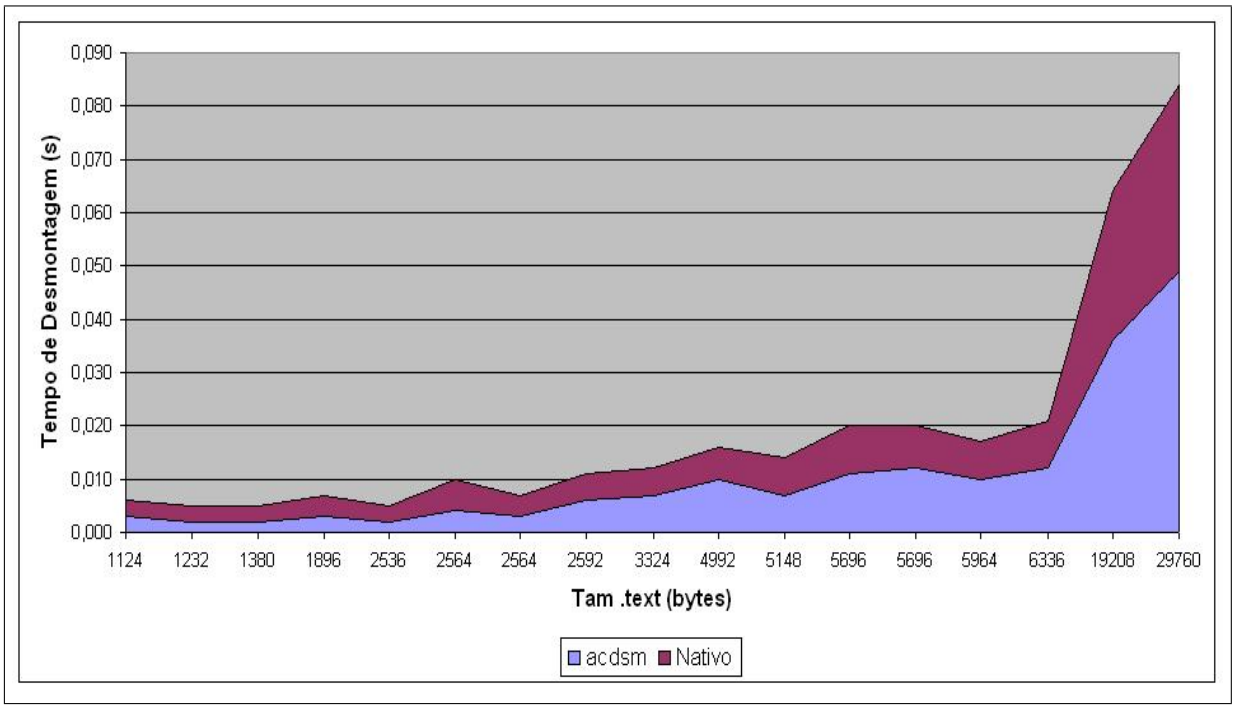


Figura 12: *Gráfico dos Resultados experimentais do acdsm para o MIPS*

			Tempo de desmontagem(s)	
Nome	Num Arqs	Tam .text(bytes)	Acdsm	Nativo
basicmath_small	4	4924	0.010	0.008
basicmath_large	4	5948	0.012	0.010
bitcount	9	4176	0.009	0.008
qsort_small	1	1060	0.003	0.003
qsort_large	1	1716	0.005	0.004
susan	1	52700	0.104	0.095
jpeg	60	228288	0.437	0.406
typeset	1	25372	0.050	0.039
dijkstra_small	1	2204	0.007	0.005
dijkstra_large	1	2204	0.005	0.006
ispell	1	1076	0.002	0.003
stringsearch_small	4	4924	0.012	0.010
stringsearch_large	4	4924	0.010	0.011
blowfish	7	15756	0.036	0.029
sha	2	2764	0.005	0.005
rawcaudio	2	2184	0.005	0.005
rawdaudio	2	2176	0.005	0.005
crc32	1	1288	0.003	0.003
fft	3	5316	0.012	0.012

Tabela 3: *Resultados experimentais do acdsm para o PowerPC*

Nome	Num Arqs	Tam .text(bytes)	Tempo de execução Acdsm (s)
int2bin	1	188	0.003
cast	1	213	0.002
sort	1	425	0.003
xram	1	214	0.002

Tabela 4: *Resultados experimentais do acdsm para o i8051*

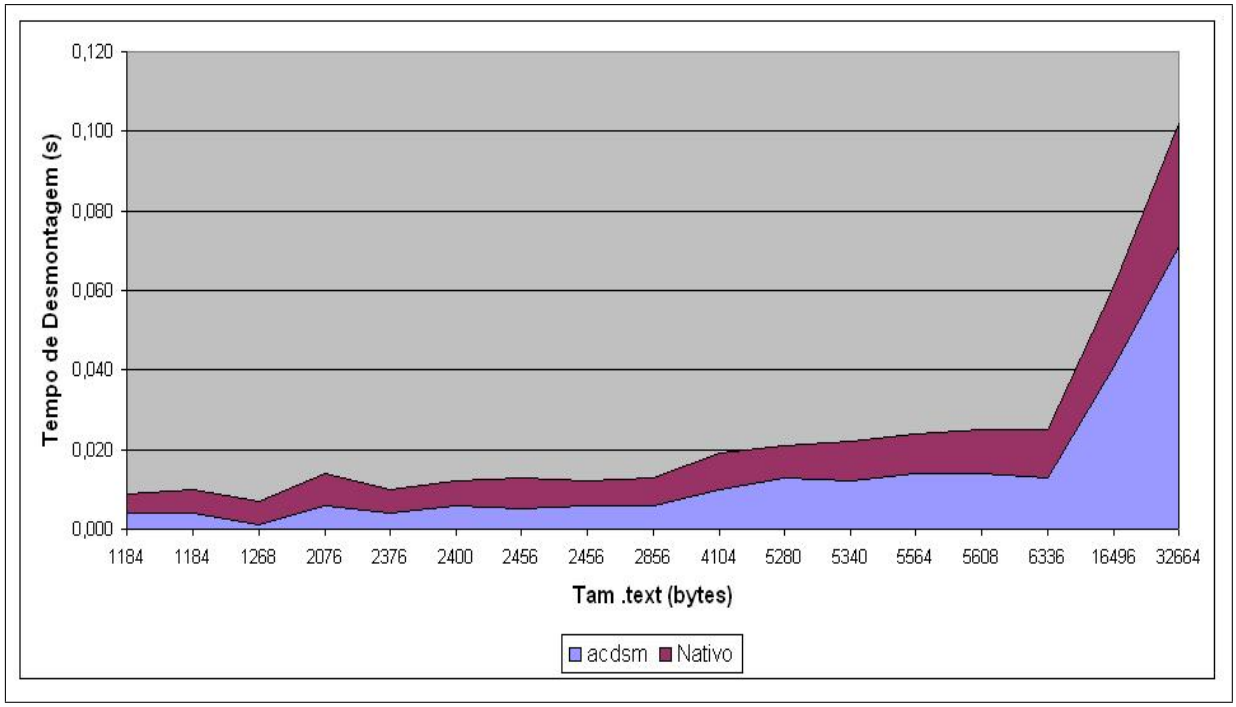


Figura 13: *Gráfico dos Resultados experimentais do acdsm para o SPARC V8*

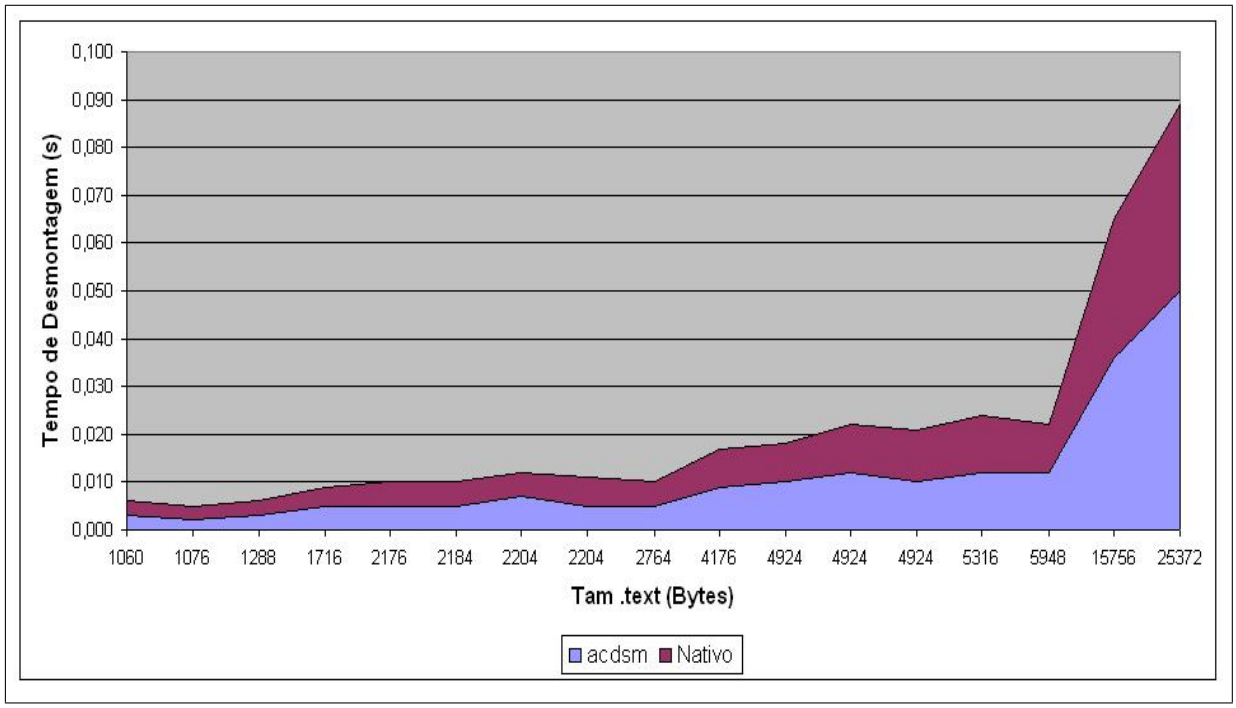


Figura 14: *Gráfico dos Resultados experimentais do acdsm para o POWERPC*

4.2 Validação do Depurador

Para a validação da ferramenta acgdb foi adotado também o benchmark Mibench rodando em três diferentes CPUs (MIPS, PowerPC, SPARC).

4.2.1 Validação das Ferramenta Geradas

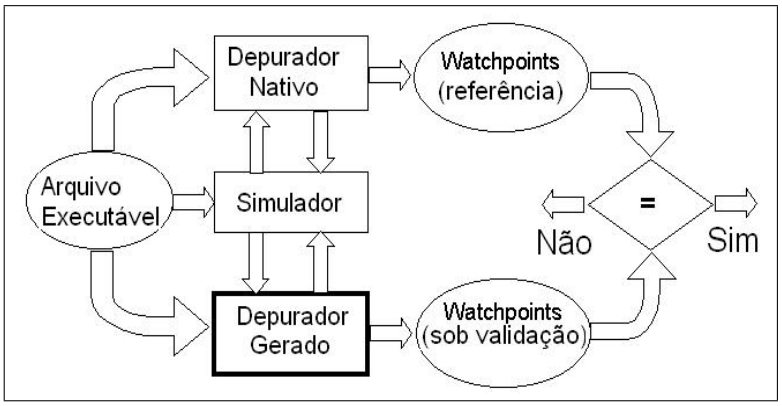


Figura 15: *Fluxo de validação dos depuradores gerados*

Para validar os depuradores gerados, utilizamos comunicação do gdb com o simulador ArchC através de uma porta serial, assim o gdb e o simulador se comunicam para a execução do programa. Foi seguido o seguinte procedimento, como visto na figura 15:

Passo 1 : Dado a descrição da CPU-alvo, foram gerados um depurador com acgdb e um simulador.

Passo 2 : Dado um arquivo objeto de entrada, este foi alimentado ao depurador e ao simulador.

Passo 3 : Foram setados breakpoints e watchpoints pelo código do programa benchmark utilizado, e submetido ao depurador, este por sua vez gerando na saída os valores dos watchpoints.

Passo 4 : Os passos 1,2,3 são repetidos para o desmontador gerado pela ferramenta acgdb e para o desmontador nativo.

Passo 5 : São comparados os valores de saída do depurador nativo e do depurador gerado por acgdb, se corresponderem a validação ocorreu corretamente.

Como todas as comparações não diferiram para cada programa do benchmark e para

cada CPU-alvo, existe uma forte evidência de validação correta.

Pode ser argumentado que tal procedimento não testa se a ferramenta funciona para os comandos de subrotinas next (passa por fora do procedimento) e step (entra no procedimento) do depurador, mas isto é justificado pelo fato dos watchpoints serem setados em variáveis locais, assim perdendo o escopo e gerando um valor errado, se o depurador não funcionar corretamente.

4.2.2 Validação da Ferramenta Geradora

Para verificar a redirecionabilidade da ferramenta geradora, o procedimento acima foi repetido para os alvos RISC (MIPS,SPARC,PowerPC), para o depurador não faz sentido termos tabelas com os resultados, pois grande parte da validação ocorre visualmente.

5 Conclusões e Trabalhos Futuros

Os geradores automáticos de desmontador e depurador parecem apropriados para dar suporte ao desenvolvimento de sistemas embutidos durante a exploração de diversas CPUs (design space exploration). Os resultados experimentais fornecem evidências da funcionalidade e da capacidade de redirecionamento das ferramentas.

5.1 Trabalhos futuros

Um dos trabalhos futuros seria a elaboração de uma API para o mecanismo de redirecionamento, permitindo assim a geração das ferramentas (montador, simulador, desmontador, depurador e outras) independentemente da ADL. E outro trabalho seria a integração do depurador com uma interface gráfica ou construção de uma, o que facilitaria em muito o uso desta ferramenta.

5.2 Produtos do trabalho

Os resultados obtidos com esse trabalho foram:

- Uma ferramenta desmontador redirecionável disponível em repositório público[www.archc.org].
- Uma ferramenta depurador redirecionável.
- Um artigo publicado no Students Forum on Microelectronics 2006 (MENDONçA et al., 2006) em co-autoria com o aluno de mestrado Max R. de O. Schultz.
- Um artigo submetido ao VLSI Design & Embedded Systems, em co-autoria com o aluno de mestrado Max R. de O. Schultz.

Referências

ABBASPOUR, M.; ZHU, J. *Retargetable Binary Utilities*.

AL, S. P. et. *LISA Machine Description Language for Cycle-Accurate Models of Programmable DSP*. 1999.

AZEVEDO, R. et al. The archc architecture description language and tools. Campinas, SP, October 2005.

BALDASSIN, A. Mestrado em Computação, *Geração Automática de Montadores em ArchC*. Campinas, SP: [s.n.], Março 2005.

BALDASSIN, A.; CENTODUCATTE, P.; RIGO, S. Extending the archc language for automatic generation of assemblers. Rio de Janeiro, RJ, October 2005.

BENCHMARK Dalton. Disponível em: <http://www.cs.ucr.edu/ dalton/i8051/i8051syn>. Acesso em: 15 de dezembro 2005.

BENCHMARK Mibench. Disponível em: <http://www.eecs.umich.edu/mibench/>. Acesso em: 15 de julho 2006.

CASAROTTO, D.; SANTOS, L. C. V. dos. Automatic link editor generation for embedded cpu cores. Florianópolis, SC, 2006.

GDB Internals. Disponível em: <http://sources.redhat.com/gdb/current/onlinedocs/gdbint$_{t}oc.html¿.Acessoem$: $22dejaneiro2006$.

GDB User Manual. Disponível em: <http://sources.redhat.com/gdb/current/onlinedocs/gdb$_{t}oc.html¿.Acessoem$: $22dejaneiro2006$.

GNU Binutils. Disponível em: <http://www.gnu.org/software/binutils>. Acesso em: 10 de julho 2005.

GUTHAUS, M. R. et al. Mibench: A free, commercially representative embedded benchmark suite. In: *International Workshop on Workload Characterization*. [S.l.: s.n.], 2001. p. 3.14.

HADJIYIANNIS, G.; HANONO, S.; DEVADAS, S. Isdl: An instruction set description language for retargetability. In: *Design Automation Conference*. [S.l.: s.n.], 1997. p. 299–302.

HARTOOG, M. R. et al. Generation of software tools from processor descriptions for hardware/software codesign. In: *Design Automation Conference*. [S.l.: s.n.], 1997. p. 303–306.

KSTNER, D. *PROPAN: A Retargetable System for Postpass Optimisations and Analyses*. 2000.

LEUPERS, R.; MARWEDEL, P. *Retargetable Compiler Technology for Embedded Systems - Tools and Applications*. [S.l.]: Kluwer Academic, 2001. ISBN 0-7923-7578-5.

MARWEDEL, P. *Embedded System Design*. [S.l.]: Kluwer Academic, 2003. 258 p. ISBN 1-4020-7690-8.

MENDONçA, A. K. I. de et al. Automatic adl-based generation of disassembling tools. Ouro Preto, MG, Agosto 2006.

PAUL, R. P. *Sparc Architecture, Assembly Language Programming, and C*. [S.l.]: Prentice Hall, 1994. ISBN 0138768897.

PESCH, R. H.; OSIER, J. M. *The GNU Binary Utilities*. maio 1993.

RIGO, S. *ArchC: Uma Linguagem de Descrição de Arquiteturas*. 105 p. Tese (Doutorado em Computação) — Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, 2004.

RIGO, S. et al. Archc: A systemc-based architecture description language. Campinas, SP, October 2004.

SALTO Project. Disponível em: <http://www.irisa.fr/caps/projects/Salto>. Acesso em: 15 de julho 2005.

TAGLIETTI, L. et al. Automatically retargetable pre-processor and assembler generation for asips. Florianópolis, SC, 2005.

ANEXO A – Código Fonte

Listing A.1: acbingen.sh

```
#!/bin/sh


#
# command line parsing
#
me=‘echo "$0" | sed -e 's,.*/,,'‘

usage="\
Usage: $0 [options] <model-file>

Create binary utilities source files and optionally build them.

Options:
  -a<name> sets the architecture name (if ommitted, it defaults to
           <model-file> without the extension
  -i<dir> build and install the binary utilities in directory <dir>
           NOTE: <dir> -MUST- be an absolute path
  -c only create the files, do not copy to binutils tree
  -h print this help
  -v print version number

Report bugs and patches to ArchC Team."

version="\
ArchC binutils generator script version 2.0beta3"
```

```
help="
Try \'$me -h for more information."

ARCH_NAME=""
CREATE_ONLY=0
BINUTILS_INST_DIR=""
PATCH_BINUTILS=1 #1 means to patch, 0 otherwise
PATCH_GDB=1 #1 means to patch, 0 otherwise
TEMP_DIR="acbingenbuilddir"

# Parse command line
while getopts ":hvca:i:" flag
do
  case $flag in
    h) echo "$usage"; exit 0 ;;
    v) echo "$version" ; exit 0 ;;
    c) CREATE_ONLY=1 ;;
    a) ARCH_NAME="$OPTARG" ;;
    i) BINUTILS_INST_DIR="$OPTARG" ;;
    *) echo "$me: invalid option $help"
       exit 1 ;;
  esac
done
shift $(($OPTIND - 1))

case $# in
 0 ) echo "No model file specified"
     exit 1 ;;
 1 ) MODEL_FILE=$1 ;;
 * ) echo "Too many arguments"
     exit 1;;
esac

if [ ! -f $MODEL_FILE ]; then
```

```
  echo "File does not exist: $MODEL_FILE"
  exit 1
fi

if [ -z "$ARCH_NAME" ]; then
  ARCH_NAME=`echo "$MODEL_FILE" | sed -e "s/\..*$//"`
fi

ARCH_INVALID_CHAR=`echo "$2" | sed -e 's/[a-zA-Z][_a-zA-Z0-9]*//'`

if [ ! -z "$ARCH_INVALID_CHAR" ]; then
  echo "Invalid architecture name: ${ARCH_NAME}"
  echo "Valid characters include letters, numbers and underscore (only"\
       "letters can begin a name)."
  exit 1
fi

if [ -z "$BINUTILS_PATH" ]; then
  BINUTILS_PATH=`grep BINUTILS_PATH /home/max/desenvolvimento/archc-prj/
      archc/branches/archc-v2.x-with-gdb-branch//etc/archc.conf | cut -d = -f2`
  BINUTILS_PATH=`echo $BINUTILS_PATH`

  if [ -z "$BINUTILS_PATH" ]; then
    echo "BINUTILS_PATH environment variable not set"
    exit 1
  fi
fi



# check for '/' at the end and take it out
BINUTILS_DIR=`echo "$BINUTILS_PATH" | sed -e 's/\/$//'`



FILES_TO_PATCH="bfd/archures.c bfd/Makefile.in bfd/bfd-in2.h bfd/config.bfd bfd/
    configure bfd/targets.c config.sub gas/configure.tgt gas/Makefile.in opcodes/
```

```
    configure opcodes/Makefile.in opcodes/disassemble.c ld/configure.tgt ld/Makefile.in
    include/dis-asm.h"
FILES_TO_COPY="gas/config/tc-xxxxx.c gas/config/tc-xxxxx.h opcodes/xxxxx-
    opc.c opcodes/xxxxx-dis.c include/opcode/xxxxx.h include/elf/xxxxx.h bfd/elf32-
    xxxxx.c bfd/cpu-xxxxx.c ld/emulparams/xxxxxelf.sh"


if [ "$CREATE_ONLY" -eq 0 ]; then
# tell the user if the architecture name already exists in Binutils
  $BINUTILS_DIR/config.sub ${ARCH_NAME}-elf > /dev/null 2>&1
  if [ $? -eq 0 ]; then
    echo "===================="
    echo "It has been detected that your Binutils distribution already uses"
    echo "the architecture name '${ARCH_NAME}'."
    echo
    echo "It is not recommended to continue if you have not set it by yourself."
# echo "If you continue, the Binutils files will not be patched."
    echo "===================="
    echo
    read -e -p "Do you wish to continue? (y) -> " USER_ANSWER; : ${
        USER_ANSWER:="y"}
    echo
    case $USER_ANSWER in
        y | yes | Y | YES) PATCH_BINUTILS=0; break;;
        * ) echo "Quitting ..."; echo; exit 1;;
    esac
  fi
fi

# creates binutils directory tree (if none was built)
mkdir -p $TEMP_DIR
[ $? -ne 0 ] && exit $?
cp -rf /home/max/desenvolvimento/archc-prj/archc/branches/archc-v2.x-with-gdb
    -branch//share/archc/binutils ${TEMP_DIR}/
```

```
# generate the files in the binutils tree
echo "Generating machine dependent code..."
/home/max/desenvolvimento/archc-prj/archc/branches/archc-v2.x-with-gdb-
    branch//bin/bmdsfg $MODEL_FILE -a${ARCH_NAME}
[ $? -ne 0 ] && exit $?

cd $TEMP_DIR

# change the name of template files
m4 -P defines.m4 binutils/ld/emulparams/xxxxxelf.sh > binutils/ld/emulparams/${
    ARCH_NAME}elf.sh
m4 -P defines.m4 binutils/include/opcode/xxxxx.h > binutils/include/opcode/${
    ARCH_NAME}.h
m4 -P defines.m4 binutils/bfd/cpu-xxxxx.c > binutils/bfd/cpu-${ARCH_NAME}.c
m4 -P defines.m4 binutils/gas/config/tc-xxxxx.h > binutils/gas/config/tc-${
    ARCH_NAME}.h
m4 -P defines.m4 binutils/opcodes/xxxxx-opc.c > binutils/opcodes/${ARCH_NAME
    }-opc.c
m4 -P defines.m4 binutils/opcodes/xxxxx-dis.c > binutils/opcodes/${ARCH_NAME
    }-dis.c
m4 -P defines.m4 binutils/include/elf/xxxxx.h > binutils/include/elf/${ARCH_NAME
    }.h
m4 -P defines.m4 binutils/bfd/elf32-xxxxx.c > binutils/bfd/elf32-${ARCH_NAME}.
    c
m4 -P defines.m4 binutils/gas/config/tc-xxxxx.c > binutils/gas/config/tc-${
    ARCH_NAME}.c


if [ "$CREATE_ONLY" -ne 0 ]; then
  echo "Done. No files copied."
  echo
  #exit 0
fi
```

```
if [ -z "$GDB_PATH" ]; then
  GDB_PATH=`grep GDB_PATH /home/max/desenvolvimento/archc-prj/archc/
      branches/archc-v2.x-with-gdb-branch//etc/archc.conf | cut -d = -f2`
  GDB_PATH=`echo $GDB_PATH`

  if [ -z "$GDB_PATH" ]; then
    echo "GDB_PATH environment variable not set"
    exit 1
  fi
fi


# check for '/' at the end and take it out
GDB_DIR=`echo "$GDB_PATH" | sed -e 's/\/$//'`


FILES_TO_PATCH_GDB="bfd/archures.c bfd/Makefile.in bfd/bfd-in2.h bfd/config.
    bfd bfd/configure bfd/targets.c config.sub gdb/configure.host gdb/configure.tgt gdb
    /Makefile.in opcodes/configure opcodes/Makefile.in opcodes/disassemble.c include/
    dis-asm.h"
FILES_TO_COPY_GDB="gdb/xxxxx-tdep.c gdb/config/xxxxx/xxxxx.mt opcodes/
    xxxxx-opc.c opcodes/xxxxx-dis.c include/opcode/xxxxx.h include/elf/xxxxx.h bfd
    /elf32-xxxxx.c bfd/cpu-xxxxx.c "


if [ "$CREATE_ONLY" -eq 0 ]; then
# tell the user if the architecture name already exists in Gdb
  $GDB_DIR/config.sub ${ARCH_NAME}-elf > /dev/null 2>&1
  if [ $? -eq 0 ]; then
    echo "==================="
    echo "It has been detected that your Gdb distribution already uses"
    echo "the architecture name '${ARCH_NAME}'."
    echo
    echo "It is not recommended to continue if you have not set it by yourself."
```

```
# echo "If you continue, the Gdb files will not be patched."
    echo "====================="
    echo
    read -e -p "Do you wish to continue? (y) -> " USER_ANSWER; : ${
        USER_ANSWER:="y"}
    echo
    case $USER_ANSWER in
        y | yes | Y | YES) PATCH_GDB=0; break;;
        * ) echo "Quitting ..."; echo; exit 1;;
    esac
  fi
fi

# creates gdb directory tree (if none was built)
cd ..
TEMP_DIR="acbingenbuilddir"
mkdir -p $TEMP_DIR
[ $? -ne 0 ] && exit $?
cp -rf /home/max/desenvolvimento/archc-prj/archc/branches/archc-v2.x-with-gdb
    -branch//share/archc/gdb ${TEMP_DIR}/



# generate the files in the gdb tree
echo "Generating machine dependent code..."
/home/max/desenvolvimento/archc-prj/archc/branches/archc-v2.x-with-gdb-
    branch//bin/bmdsfg $MODEL_FILE -a${ARCH_NAME}
[ $? -ne 0 ] && exit $?

cd $TEMP_DIR
mkdir -p gdb/gdb/config/${ARCH_NAME}

# change the name of template files
m4 -P defines.m4 gdb/include/opcode/xxxxx.h > gdb/include/opcode/${
    ARCH_NAME}.h
m4 -P defines.m4 gdb/bfd/cpu-xxxxx.c > gdb/bfd/cpu-${ARCH_NAME}.c
```

```
m4 -P defines.m4 gdb/opcodes/xxxxx-opc.c > gdb/opcodes/${ARCH_NAME}-opc.c
m4 -P defines.m4 gdb/opcodes/xxxxx-dis.c > gdb/opcodes/${ARCH_NAME}-dis.c
m4 -P defines.m4 gdb/include/elf/xxxxx.h > gdb/include/elf/${ARCH_NAME}.h
m4 -P defines.m4 gdb/bfd/elf32-xxxxx.c > gdb/bfd/elf32-${ARCH_NAME}.c
m4 -P defines.m4 gdb/gdb/xxxxx-tdep.c > gdb/gdb/${ARCH_NAME}-tdep.c
m4 -P defines.m4 gdb/gdb/config/xxxxx/xxxxx.mt > gdb/gdb/config/${
    ARCH_NAME}/${ARCH_NAME}.mt


if [ "$CREATE_ONLY" -ne 0 ]; then
  echo "Done. No files copied."
  echo
  exit 0
fi




if [ "$PATCH_BINUTILS" -ne 0 ]; then
# applies the patch
    echo "Patching... "
    for file in $FILES_TO_PATCH
    do

      if [ ! -f "$BINUTILS_DIR/$file" ]; then

          if [ $file = "gas/configure.tgt" ]; then
            # Starting from 2.16, binutils uses configure.tgt
            # For compatibility with older versions, redirect to 'configure'
            file="gas/configure"
            cp -f binutils/gas/configure.tgt.sed binutils/gas/configure.sed > /dev/
                null 2>&1
          else
            echo "Source file $file not found."
            exit 1
```

```
        fi
      fi

  # creates a backup
      mv -f $BINUTILS_DIR/$file $BINUTILS_DIR/$file.bkp > /dev/null 2>&1
      if [ $? -ne 0 ]; then
          echo "Cannot move file $file."
          exit 1
      fi

      sed s/xxxxx/${ARCH_NAME}/g binutils/$file.sed > binutils/$file.sed2
      sed -f binutils/$file.sed2 $BINUTILS_DIR/$file.bkp > $BINUTILS_DIR/$file

    done

    chmod a+x $BINUTILS_DIR/config.sub
    chmod a+x $BINUTILS_DIR/bfd/config.bfd

else
    echo "Skipping patching..."
fi

# copies the generated files into binutils tree
echo "Copying files to binutils source tree..."
for file in $FILES_TO_COPY
do
  cpfile=`echo "$file" | sed -e "s/xxxxx/${ARCH_NAME}/g"`
  cp -f binutils/${cpfile} $BINUTILS_DIR/${cpfile}
  [ $? -ne 0 ] && exit $?
done




if [ "$PATCH_GDB" -ne 0 ]; then
```

```
# applies the patch
    echo "Patching... "
    for file in $FILES_TO_PATCH_GDB
    do

      if [ ! -f "$GDB_DIR/$file" ]; then

            if [ $file = "gdb/configure.tgt" ]; then
              # Starting from 2.16, gdb uses configure.tgt
              # For compatibility with older versions, redirect to 'configure'
              file="gdb/configure"
              cp -f gdb/gdb/configure.tgt.sed gdb/gdb/configure.sed > /dev/null
                  2>&1
            else
              echo "Source file $file not found."
              exit 1
        fi
      fi

  # creates a backup
      mv -f $GDB_DIR/$file $GDB_DIR/$file.bkp > /dev/null 2>&1
      if [ $? -ne 0 ]; then
          echo "Cannot move file $file."
          exit 1
      fi

      sed s/xxxxx/${ARCH_NAME}/g gdb/$file.sed > gdb/$file.sed2
      sed -f gdb/$file.sed2 $GDB_DIR/$file.bkp > $GDB_DIR/$file

    done

    chmod a+x $GDB_DIR/config.sub
    chmod a+x $GDB_DIR/bfd/config.bfd

else
```

```
    echo "Skipping patching..."
fi

# copies the generated files into gdb tree
mkdir -p $GDB_DIR/gdb/config/${ARCH_NAME}
echo "Copying files to gdb source tree..."
for file in $FILES_TO_COPY_GDB
do
  cpfile=`echo "$file" | sed -e "s/xxxxx/${ARCH_NAME}/g"`
  cp -f gdb/${cpfile} $GDB_DIR/${cpfile}
  [ $? -ne 0 ] && exit $?
done




if [ ! -z ${BINUTILS_INST_DIR} ]; then
  echo "Installing the binary utilities..."

  mkdir -p build
  cd build
  $BINUTILS_DIR/configure --target=${ARCH_NAME}-elf --prefix=`pwd` --
      bindir=${BINUTILS_INST_DIR}
  [ $? -ne 0 ] && exit $?
  make
  [ $? -ne 0 ] && exit $?
  make install
  [ $? -ne 0 ] && exit $?



  cd ..
  mkdir -p build_gdb
  cd build_gdb
  $GDB_DIR/configure --target=${ARCH_NAME}-elf --prefix=`pwd` --bindir=$
      {BINUTILS_INST_DIR}
```

```
  [ $? -ne 0 ] && exit $?
  make
  [ $? -ne 0 ] && exit $?
  make install
  [ $? -ne 0 ] && exit $?

fi

echo "All done successfully."
echo
```

Listing A.2: bfd.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file bfd.c
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC-UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE-UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 -0300
 *
 * @brief BFD library related code (implementation)
```

```
 *
 * @attention Copyright (C) 2002--2006 --- The ArchC Team
 *
 */

#include <stdlib.h>
#include <stdio.h>

#include "utils.h"
#include "bfd.h"

// Linker generation definitions...

typedef struct _ac_relocation_type {
  char *name;
  unsigned id;
  unsigned rightshift;
  unsigned reloc_size;
  unsigned bitsize;
  unsigned bitpos;
  unsigned mask;
  unsigned pc_relative;
  unsigned uses_carry;
  struct _ac_relocation_type *next;
} ac_relocation_type;

static ac_relocation_type* find_relocation_by_id(unsigned id);
static void process_instruction_relocation(ac_asm_insn *asml);
static ac_relocation_type* find_relocation(ac_relocation_type* reloc);
static int compare_relocs(ac_relocation_type* first, ac_relocation_type* second);



static ac_relocation_type *relocation_types_list = NULL;
```

```
void create_relocation_list()
{
  ac_asm_insn *asml = ac_asm_get_asm_insn_list();

  while (asml != NULL) {
    if (asml->insn != NULL) { /* native instructions */
      process_instruction_relocation(asml);
    }
    asml = asml->next;
  }

}

/*
  Relocation IDs - include/elf/<arch>.h
*/
int CreateRelocIds(const char *relocid_filename)
{
  FILE *output;

  if ((output = fopen(relocid_filename, "w")) == NULL)
    return 0;

  fprintf(output, "%sRELOC_NUMBER (R_%s_NONE, 0)\n", IND1, get_arch_name());

  unsigned reloc_id = 1;
  ac_relocation_type* relocation = find_relocation_by_id(reloc_id);
  while (relocation) {
    fprintf (output, "%sRELOC_NUMBER (%s, %d)\n", IND1, relocation->name,
        relocation->id);
    reloc_id++;
    relocation = find_relocation_by_id(reloc_id);
  }
```

```
  // Generic data relocations
  fprintf(output, "%sRELOC_NUMBER (R_%s_8,  %d)\n", IND1, get_arch_name(),
      reloc_id);
  fprintf(output, "%sRELOC_NUMBER (R_%s_16, %d)\n", IND1, get_arch_name(),
      reloc_id + 1);
  fprintf(output, "%sRELOC_NUMBER (R_%s_32, %d)\n", IND1, get_arch_name(),
      reloc_id + 2);

  fprintf(output, "%sRELOC_NUMBER (R_%s_REL8,  %d)\n", IND1, get_arch_name
      (), reloc_id + 3);
  fprintf(output, "%sRELOC_NUMBER (R_%s_REL16, %d)\n", IND1, get_arch_name
      (), reloc_id + 4);
  fprintf(output, "%sRELOC_NUMBER (R_%s_REL32, %d)", IND1, get_arch_name(),
      reloc_id + 5);

  fclose(output);
  return 1;
}

/*
  Relocation HOWTO structure - bfd/elf32-<arch>.c
*/
int CreateRelocHowto(const char *reloc_howto_filename)
{
  FILE *output;

  if ((output = fopen(reloc_howto_filename, "w")) == NULL)
    return 0;

  fprintf(output, "%sHOWTO (R_%s_NONE, 0, 0, 0, FALSE, 0,
      complain_overflow_dont, bfd_elf_archc_reloc, \"R_%s_NONE\", FALSE, 0, 0,
      FALSE),\n", IND1, get_arch_name(), get_arch_name());

  unsigned reloc_id = 1;
  ac_relocation_type *relocation = find_relocation_by_id(reloc_id);
```

```
while (relocation) {
  char relocation_function[256];
  sprintf(relocation_function, "_bfd_%s_elf_carry_reloc", get_arch_name());

  fprintf (output, "%sHOWTO␣(%s,␣%d,␣%d,␣%d,␣%s,␣%d,␣%s,␣%s,␣\"%s\",␣%s,␣0x
      %x,␣0x%x,␣%s),\n",
    IND1,
    relocation->name,
    relocation->rightshift,
    relocation->reloc_size,
    relocation->bitsize,
    relocation->pc_relative ? "TRUE" : "FALSE",
    relocation->bitpos,
    "complain_overflow_dont",
    relocation->uses_carry ? relocation_function : "bfd_elf_archc_reloc",
    relocation->name,
    "FALSE", // PENDENTE. Isso mesmo?
    0, // PENDENTE - Usar o mesmo do de baixo ou 0?
    relocation->mask,
    "TRUE" // PENDENTE. Isso mesmo?
  );
  reloc_id++;
  relocation = find_relocation_by_id(reloc_id);
}
// Generic data relocations
fprintf(output, "%sHOWTO␣(R_%s_8,␣␣0,␣8,␣␣8,␣FALSE,␣0,␣
    complain_overflow_bitfield,␣bfd_elf_archc_reloc,␣␣\"R_%s_8\",␣␣␣␣FALSE,␣0,␣0
    x000000ff,␣TRUE),\n", IND1, get_arch_name(), get_arch_name());
fprintf(output, "%sHOWTO␣(R_%s_16,␣0,␣16,␣16,␣FALSE,␣0,␣
    complain_overflow_bitfield,␣bfd_elf_archc_reloc,␣␣\"R_%s_16\",␣␣␣FALSE,␣0,␣0
    x0000ffff,␣TRUE),\n", IND1, get_arch_name(), get_arch_name());
fprintf(output, "%sHOWTO␣(R_%s_32,␣0,␣32,␣32,␣FALSE,␣0,␣
    complain_overflow_bitfield,␣bfd_elf_archc_reloc,␣␣\"R_%s_32\",␣␣␣FALSE,␣0,␣0
    xffffffff,␣TRUE),\n", IND1, get_arch_name(), get_arch_name());
```

```
  fprintf(output, "%sHOWTO (R_%s_REL8,  0, 8,  8, TRUE, 0,
      complain_overflow_bitfield, bfd_elf_archc_reloc,  \"R_%s_REL8\",    FALSE, 0, 0
      x000000ff, TRUE),\n", IND1, get_arch_name(), get_arch_name());
  fprintf(output, "%sHOWTO (R_%s_REL16, 0, 16, 16, TRUE, 0,
      complain_overflow_bitfield, bfd_elf_archc_reloc,  \"R_%s_REL16\",   FALSE, 0, 0
      x0000ffff, TRUE),\n", IND1, get_arch_name(), get_arch_name());
  fprintf(output, "%sHOWTO (R_%s_REL32, 0, 32, 32, TRUE, 0,
      complain_overflow_bitfield, bfd_elf_archc_reloc,  \"R_%s_REL32\",   FALSE, 0, 0
      xffffffff, TRUE),\n", IND1, get_arch_name(), get_arch_name());

  fclose(output);
  return 1;
}


/*
  Relocation map - bfd/elf32-<arch>.c
*/
int CreateRelocMap(const char *relocmap_filename)
{
  FILE *output;

  if ((output = fopen(relocmap_filename, "w")) == NULL)
    return 0;

  fprintf(output,"%s{ BFD_RELOC_NONE, R_%s_NONE },\n",IND1, get_arch_name()
      );

  unsigned reloc_id = 1;
  ac_relocation_type *relocation = find_relocation_by_id(reloc_id);
  while (relocation) {
    reloc_id++;
    ac_relocation_type* next_relocation = find_relocation_by_id(reloc_id);
    fprintf(output,"%s{ %d, %s },\n", IND1, relocation->id, relocation->name);
    relocation = next_relocation;
```

```
  }

  // Generic data relocations
  fprintf(output,"%s{ BFD_GENERIC_8,  R_%s_8  },\n",IND1, get_arch_name());
  fprintf(output,"%s{ BFD_GENERIC_16, R_%s_16 },\n",IND1, get_arch_name());
  fprintf(output,"%s{ BFD_GENERIC_32, R_%s_32 },\n",IND1, get_arch_name());

  fprintf(output,"%s{ BFD_GENERIC_REL8,  R_%s_REL8  },\n",IND1,
      get_arch_name());
  fprintf(output,"%s{ BFD_GENERIC_REL16, R_%s_REL16 },\n",IND1,
      get_arch_name());
  fprintf(output,"%s{ BFD_GENERIC_REL32, R_%s_REL32 }\n", IND1,
      get_arch_name());

  fclose(output);
  return 1;
}



static ac_relocation_type* find_relocation_by_id(unsigned id)
{
  ac_relocation_type* relocation = relocation_types_list;
  while (relocation) {
    if (relocation->id == id)
      return relocation;
    relocation = relocation->next;
  }

  return NULL;
}



static void process_instruction_relocation(ac_asm_insn *asml)
{
```

```
ac_operand_list *opP = asml->operands;

while (opP != NULL) { /* for each operand */

  if (opP->type != op_addr && opP->type != op_exp) {
     opP = opP->next;
     continue;
  }

  ac_asm_insn_field *fieldP = opP->fields;

   /* Get the sum of all fields in bits (from left to right)
    */
  unsigned int fields_bit_size = 0;
  for (; fieldP != NULL; fieldP = fieldP->next)
    fields_bit_size += fieldP->size;

  fieldP = opP->fields;
  while (fieldP != NULL) { /* for each field */

    fields_bit_size -= fieldP->size;

    /* fill in the relocation fields */

    /*
     * rightshift
     * - number of bits the reloc value will be shifted to the right
     *
     * if the operand has multiple fields assigned to, then the
     * left fields must be shifted to the right
     */
    unsigned rightshift = 0+fields_bit_size;

    /*
     * reloc_size
```

```
 * − size of the relocation field
 * it is the size of the instruction
 *
 * NOTE: Since we've made our own relocation routines, this value
 * is being used as the bit size
 */
//unsigned reloc_size = log_table[get_insn_size(asml)/8];
unsigned reloc_size = get_insn_size(asml);

/*
 * bitsize
 * − size (in bits) of the field to be relocated
 */
unsigned bitsize = fieldP->size;



/*
 * bitpos
 * − bit position of the relocation value in the destination
 * the relocation is shifted left by the number of bits
 * specified in this variable
 */
unsigned bitpos = get_insn_size(asml) − (fieldP->first_bit+1);

/*
 * mask
 * − selects which parts of the relocation field is inserted
 * into the binary image
 */
unsigned long mask = 0;
unsigned i;
for (i=0; i < get_insn_size(asml); i++)
  mask |= (1 << i);
```

```
mask >>= (get_insn_size(asml) − fieldP−>size);

 /*
  * uses_carry
  * − if the carry of the lower part affect the high part of
  * a field, then this flag should be 1
  */
unsigned uses_carry = 0;

/*
 * is_pcrel
 * − 1 if the field is pc−relative
 */
unsigned is_pcrel = 0;

/*
 * has_high
 * − 1 if the operand has a high modifier assigned
 *
 */
unsigned has_high = 0;

 /*
  * now apply the modifiers to the relocation fields
  */
ac_modifier_list *modP = opP−>modifiers;
while (modP != NULL) {

  switch (modP−>type) {
    case mod_low:
      if (modP−>addend >= 0 && modP−>addend <= fieldP−>size) {
        mask >>= (fieldP−>size − modP−>addend);
        bitsize = modP−>addend;
      }
      /* signed / unsigned − not being used! */
```

```
        break;

      case mod_high:
        if (modP->addend >= 0 && modP->addend <= fieldP->size) {
          rightshift += get_insn_size(asml) - modP->addend;
          mask >>= (fieldP->size - modP->addend);
          bitsize = modP->addend;
        }
        else
          rightshift += get_insn_size(asml) - fieldP->size;

        uses_carry = modP->carry;
        has_high = 1;

        /* signed / unsigned - not being used! */
        break;

      case mod_aligned:
          if (modP->addend == -1)
              modP->addend = get_arch_size()/8;
          rightshift += log_table[modP->addend];
        break;

      case mod_pcrel:
        is_pcrel = 1;
        break;

      case mod_pcrelext:
        is_pcrel = 1;
        break;
    }

    modP = modP->next;
  }
```

```
      mask <<= bitpos;

      ac_relocation_type* reloc = (ac_relocation_type*) malloc(sizeof(ac_relocation_type
          ));
      reloc->rightshift = rightshift;
      reloc->reloc_size = reloc_size;
      reloc->bitsize = bitsize;
      reloc->pc_relative = is_pcrel;
      reloc->bitpos = bitpos;
      reloc->mask = mask;
      reloc->uses_carry = uses_carry;

      ac_relocation_type* reloc_found = find_relocation(reloc);

      if (!reloc_found) {
        reloc->id = relocation_types_list ? relocation_types_list->id + 1 : 1;
        reloc->name = (char*) malloc(sizeof(char)*128);
        sprintf(reloc->name, "R_%s_%d_%s%s%d%s", get_arch_name(), reloc->id, (
            is_pcrel ? "REL" : "") , (has_high ? "HI" : "LO"), bitsize, uses_carry ? "
            _CARRY" : "");

        reloc->next = relocation_types_list;
        relocation_types_list = reloc;
        fieldP->reloc_id = reloc->id;
      }
      else {
        fieldP->reloc_id = reloc_found->id;
        free(reloc);
      }

      fieldP = fieldP->next;
    }

    opP = opP->next;
  }
```

```
}



static ac_relocation_type* find_relocation(ac_relocation_type* reloc)
{
  ac_relocation_type* relocation = relocation_types_list;

  while (relocation != NULL) {
    if (compare_relocs(reloc,relocation))
      return relocation;
    relocation = relocation->next;
  }

  return NULL;
}

static int compare_relocs(ac_relocation_type* first, ac_relocation_type* second)
{
  return first->rightshift == second->rightshift &&
         first->reloc_size == second->reloc_size &&
         first->bitsize == second->bitsize &&
         first->bitpos == second->bitpos &&
         first->mask == second->mask &&
         first->pc_relative == second->pc_relative &&
         first->uses_carry == second->uses_carry;
}
```

Listing A.3: bfd.h

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file bfd.h
 * @author Alexandro Baldassin (UNICAMP)
```

```
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC-UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE-UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 -0300
 *
 * @brief BFD library related code
 *
 * The BFD module deals with the aspects involved in retargeting
 * the BFD library.
 *
 * @attention Copyright (C) 2002-2006 --- The ArchC Team
 *
 */

/** @defgroup bfd_group BFD library
 * @ingroup binutils_group
 *
 * @{
 */

#ifndef _BFD_H_
#define _BFD_H_
```

```
extern void create_relocation_list();
extern int CreateRelocIds(const char *relocid_filename);
extern int CreateRelocHowto(const char *reloc_howto_filename);
extern int CreateRelocMap(const char *relocmap_filename);


#endif /* _BFD_H_ */

/* @} */
```

Listing A.4: gas.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file gas.c
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC-UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE-UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 -0300
 *
 * @brief GNU assembler related code (implementation)
 *
```

```
 * @attention Copyright (C) 2002-2006 --- The ArchC Team
 *
 */

#include <stdio.h>
#include "utils.h"
#include "gas.h"


int CreateEncodingFunc(const char *encfunc_filename)
{ /* only works with 32-bit max */
  FILE *output;

  if ((output = fopen(encfunc_filename, "w")) == NULL)
    return 0;

  fprintf(output, "%sunsigned long mask1 = 0xffffffff;\n", IND1);
  fprintf(output, "%sunsigned long mask2 = 0xffffffff;\n\n", IND1);

  /* instruction format type switch */
  fprintf(output, "%sswitch(insn_type) {\n", IND1);

  ac_dec_format *pfrm = format_ins_list;
  ac_dec_field *pf = NULL;

  int i=0;
  while (pfrm != NULL) {
    fprintf(output, "%scase %i:\n", IND2, i);
    fprintf(output, "%smask2 >>= 32-%d;\n", IND3, pfrm->size);

    /* field number switch */
    fprintf(output, "%sswitch (field_id) {\n", IND3);

    pf = pfrm->fields;
    int j=0;
```

```
    while (pf != NULL) {
      fprintf(output, "%scase␣%i:\n", IND4, j);
      fprintf(output, "%smask1␣<<=␣%i;\n", IND4, pfrm->size - (pf->first_bit+1))
          ;
      fprintf(output, "%smask2␣>>=␣%i;\n", IND4, pf->first_bit+1 - pf->size);
      fprintf(output, "%sreturn␣(value␣<<␣(%i))␣&␣(mask1␣&␣mask2);\n\n", IND4,
          pfrm->size - (pf->first_bit+1));

      pf = pf->next;
      j++;
    }
    fprintf(output, "%s}\n", IND3);

    fprintf(output, "\n%sbreak;\n", IND3);
    pfrm = pfrm->next;
    i++;
  }
  fprintf(output, "\n%s}\n", IND1);

  fprintf(output, "%sreturn␣0;", IND1);

  fclose(output);
  return 1;
}



int CreateGetFieldSizeFunc(const char *getfsz_filename)
{
  FILE *output;

  if ((output = fopen(getfsz_filename, "w")) == NULL)
    return 0;

  /* instruction type switch */
```

```
  fprintf(output, "%sswitch(insn_fmt) {\n", IND1);

  ac_dec_format *pfrm = format_ins_list;
  ac_dec_field *pf = NULL;

  int i=0;
  while (pfrm != NULL) {
    fprintf(output, "%scase %i:\n", IND2, i);

    /* field number switch */
    fprintf(output, "%sswitch (field_id) {\n", IND3);

    pf = pfrm->fields;
    int j=0;
    while (pf != NULL) {
      fprintf(output, "%scase %i:\n", IND4, j);
      fprintf(output, "%sreturn %i;\n\n", IND4, pf->size);

      pf = pf->next;
      j++;
    }
    fprintf(output, "%s}\n", IND3);

    fprintf(output, "\n%sbreak;\n", IND3);
    pfrm = pfrm->next;
    i++;
  }
  fprintf(output, "\n%s}\n", IND1);

  fprintf(output, "%sreturn 0;", IND1);

  fclose(output);
  return 1;
}
```

```
int CreateGetInsnSizeFunc(const char *insnsz_filename)
{
  FILE *output;

  if ((output = fopen(insnsz_filename, "w")) == NULL)
    return 0;

  /* instruction type switch */
  fprintf(output, "%sswitch(insn_fmt) {\n", IND1);

  ac_dec_format *pfrm = format_ins_list;

  int i=0;
  while (pfrm != NULL) {
    fprintf(output, "%scase %i:\n", IND2, i);

    fprintf(output, "%sreturn %d;\n", IND3, pfrm->size);

    pfrm = pfrm->next;
    i++;
  }
  fprintf(output, "\n%s}\n", IND1);

  fprintf(output, "%sreturn 0;", IND1);

  fclose(output);
  return 1;
}
```

Listing A.5: gas.h

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file gas.h
```

```
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC-UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE-UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 -0300
 *
 * @brief GNU assembler related code
 *
 * GAS module generates target-specific code to retarget the
 * GNU assembler.
 *
 * @attention Copyright (C) 2002-2006 --- The ArchC Team
 *
 */


/** @defgroup gas_group GNU Assembler
 * @ingroup binutils_group
 *
 * @{
 */

#ifndef _GAS_H_
```

```
#define _GAS_H_

extern int CreateEncodingFunc(const char *encfunc_filename);
extern int CreateGetFieldSizeFunc(const char *getfsz_filename);
extern int CreateGetInsnSizeFunc(const char *insnsz_filename);


#endif /* _GAS_H_ */

/* @} */
```

Listing A.6: main.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file main.c
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC-UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE-UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 -0300
 *
 * @brief ArchC binary utilities generator (main file)
```

```
 *
 * This file drives the generation of the GNU binutils target-dependent files.
 * The main libraries and tools are: BFD, Opcodes, gas, ld, objdump
 *
 * @attention Copyright (C) 2002-2006 --- The ArchC Team
 *
 */

/** @defgroup binutils_group The ArchC binary utilities generator
 *
 * The ArchC binutils generator is composed of severals sub-modules, each
 * dealing with a specific component of files generation.
 *
 * @{
 */

#include <stdlib.h>
#include <stdio.h>
#include <getopt.h>

#include "utils.h"
#include "opcodes.h"
#include "bfd.h"
#include "gas.h"


static int Createm4File();


/*
 * File names definition
 */
#define OPCODE_TABLE_FILE "opcode.table"
#define SYMBOL_TABLE_FILE "symbol.table"
#define PSEUDO_TABLE_FILE "pseudo.table"
```

```
#define RELOC_IDS_FILE "reloc.ids"
#define RELOC_HOWTO_FILE "reloc.howto"
#define RELOC_MAP_FILE "reloc.map"
#define ENCODING_FN_FILE "encoding.fn"
#define FIELD_SIZE_FN_FILE "fieldsize.fn"
#define INSN_SIZE_FN_FILE "insnsize.fn"

#define GEN_DIR "acbingenbuilddir/"



/*
 Command line parsing stuff
*/
/* Show usage function (--help) */
static void show_usage(FILE * stream)
{
  fprintf(stream, "This is bmdsfg, the Binutils Machine-Dependent Source File
      Generator.\n\n");
  fprintf(stream, "Usage: bmdsfg [options] file...\n");
  fprintf(stream, "Options:\n");

  fprintf(stream, "Mandatory:\n");
  fprintf(stream, "\
  -a, --arch=<arch_name>     Set the name of the architecture to be \n\
                             built to <arch_name>\n");

  fprintf(stream, "Optional:\n");
  fprintf(stream, "\
  -h, --help                 Display this information\n");
  fprintf(stream, "\
  -v, --version              Display bmdsfg version number\n");



  fprintf(stream, "\nReport bugs to ArchC Team: www.archc.org\n");
}
```

```
/* Show version number (--version) */
static void show_version(FILE *stream)
{
  fprintf(stream, "This is bmdsfg version %s\n", ACVERSION);
}


static const char *shortopts = "-a:hv";

static const struct option longopts[] =
{
  {"help", no_argument, NULL, 'h'},
  {"version", no_argument, NULL, 'v'},
  {"arch", required_argument, NULL, 'a'},
  {NULL, no_argument, NULL, 0}
};


char *file_name = NULL; /* name of the main ArchC file */

/*
  Main Code

*/
int main(int argc, char **argv)
{
  char buffer[200];

  /* Initializes the pre-processor */
  acppInit(1);

  /* Command line parsing code */
  while (1) {
```

```
int longind;
int optc = getopt_long_only(argc, argv, shortopts, longopts,
                &longind);

if (optc == -1)
  break;

switch (optc) {

case 1: /* file */
  if (file_name != NULL) {
    fprintf(stderr, "Argument duplicated: '%s'.\n", optarg);
    exit(1);
  }
  file_name = (char *) malloc(strlen(optarg)+1);
  strcpy(file_name, optarg);
  break;

case 'a':
  if (get_arch_name() != NULL) {
    fprintf(stderr, "Argument duplicated: '%s'.\n", optarg);
    exit(1);
  }
  set_arch_name(optarg);
  break;

case 'h':
  show_usage(stdout);
  exit(0);

case 'v':
  show_version(stdout);
  exit(0);

default:
```

```
      show_usage(stdout);
      exit(1);
    }
  }

  if (file_name == NULL) {
    fprintf(stderr, "No ArchC description file specified.\n");
    exit(1);
  }

  if (get_arch_name() == NULL) {
    fprintf(stderr, "No architecture name specified.\n");
    exit(1);
  }

  /* Parse the ARCH file */
  if (!acppLoad(file_name)) {
    fprintf(stderr, "Invalid file: '%s'.\n", file_name);
    exit(1);
  }

  if (acppRun()) {
    fprintf(stderr, "Parser error in ARCH file.\n");
    exit(1);
  }
  acppUnload();



  /* Parse the ISA file */
  if (!acppLoad(isa_filename)) {
    fprintf(stderr, "Invalid ISA file: '%s'.\n", isa_filename);
    exit(1);
  }
  if (acppRun()) {
    fprintf(stderr, "Parser error in ISA file.\n");
```

```
  exit(1);
}
acppUnload();



/*
  File Generation
*/

if (!Createm4File()) {
  fprintf(stderr, "Error creating m4 file.\n");
  exit(1);
}

// Create the relocation list first
create_relocation_list();

strcpy(buffer, GEN_DIR);
strcat(buffer, OPCODE_TABLE_FILE);
if (!CreateOpcodeTable(buffer)) { /* write the opcode table */
  fprintf(stderr, "Error creating opcode table.\n");
  exit(1);
}

strcpy(buffer, GEN_DIR);
strcat(buffer, SYMBOL_TABLE_FILE);
if (!CreateAsmSymbolTable(buffer)) { /* write the symbol table */
  fprintf(stderr, "Error creating symbol table.\n");
  exit(1);
}

strcpy(buffer, GEN_DIR);
strcat(buffer, PSEUDO_TABLE_FILE);
if (!CreatePseudoOpsTable(buffer)) { /* write the pseudo-op table */
  fprintf(stderr, "Error creating pseudo table.\n");
```

```
    exit(1);
  }

  strcpy(buffer, GEN_DIR);
  strcat(buffer, RELOC_IDS_FILE);
  if (!CreateRelocIds(buffer)) {
    fprintf(stderr, "Error creating relocation IDs.\n");
    exit(1);
  }

  strcpy(buffer, GEN_DIR);
  strcat(buffer, RELOC_HOWTO_FILE);
  if (!CreateRelocHowto(buffer)) {
    fprintf(stderr, "Error creating relocation HOWTO structure.\n");
    exit(1);
  }

  strcpy(buffer, GEN_DIR);
  strcat(buffer, RELOC_MAP_FILE);
  if (!CreateRelocMap(buffer)) {
    fprintf(stderr, "Error creating relocation map.\n");
    exit(1);
  }

  strcpy(buffer, GEN_DIR);
  strcat(buffer, ENCODING_FN_FILE);
  if (!CreateEncodingFunc(buffer)) {
    fprintf(stderr, "Error creating encoding function.\n");
    exit(1);
  }

  strcpy(buffer, GEN_DIR);
  strcat(buffer, FIELD_SIZE_FN_FILE);
  if (!CreateGetFieldSizeFunc(buffer)) {
    fprintf(stderr, "Error creating field size function.\n");
```

```
    exit(1);
  }

  strcpy(buffer, GEN_DIR);
  strcat(buffer, INSN_SIZE_FN_FILE);
  if (!CreateGetInsnSizeFunc(buffer)) {
    fprintf(stderr, "Error creating insn size function.\n");
    exit(1);
  }


  return 0;
}


static int Createm4File()
{
  FILE *output;

  if ((output = fopen("acbingenbuilddir/defines.m4", "w")) == NULL)
    return 0;

  /* disable comments */
  fprintf(output, "m4_changecom()m4_dnl\n");

  fprintf(output, "m4_define(`___arch_name___',`%s')m4_dnl\n", get_arch_name());
  fprintf(output, "m4_define(`___word_size___',`%d')m4_dnl\n", get_arch_size());
  fprintf(output, "m4_define(`___max_format_size___',`%d')m4_dnl\n",
      get_max_format_size());
  fprintf(output, "m4_define(`___variable_format_size___',`%d')m4_dnl\n",
      get_variable_format_size());

  /* 1 = big, 0 = little */
  fprintf(output, "m4_define(`___endian_str___',`%s')m4_dnl\n", ac_tgt_endian ? "
      AC_BIG_ENDIAN" : "AC_LITTLE_ENDIAN");
```

```
  fprintf(output, "m4_define(‘___endian_val___’,␣‘%d’)m4_dnl\n", ac_tgt_endian ? 1 : 0);


  fprintf(output, "m4_define(‘___opcode_table___’,␣‘m4_include(%s)’)m4_dnl\n",
      OPCODE_TABLE_FILE);
  fprintf(output, "m4_define(‘___symbol_table___’,␣‘m4_include(%s)’)m4_dnl\n",
      SYMBOL_TABLE_FILE);
  fprintf(output, "m4_define(‘___pseudo_table___’,␣‘m4_include(%s)’)m4_dnl\n",
      PSEUDO_TABLE_FILE);


  fprintf(output, "m4_define(‘___reloc_ids___’,␣‘m4_include(%s)’)m4_dnl\n",
      RELOC_IDS_FILE);
  fprintf(output, "m4_define(‘___reloc_howto___’,␣‘m4_include(%s)’)m4_dnl\n",
      RELOC_HOWTO_FILE);
  fprintf(output, "m4_define(‘___reloc_map___’,␣‘m4_include(%s)’)m4_dnl\n",
      RELOC_MAP_FILE);


  fprintf(output, "m4_define(‘___encoding_function___’,␣‘m4_include(%s)’)m4_dnl\n",
      ENCODING_FN_FILE);
  fprintf(output, "m4_define(‘___fieldsize_function___’,␣‘m4_include(%s)’)m4_dnl\n",
      FIELD_SIZE_FN_FILE);
  fprintf(output, "m4_define(‘___insnsize_function___’,␣‘m4_include(%s)’)m4_dnl\n",
      INSN_SIZE_FN_FILE);

  fclose(output);

  return 1;
}

/** @} */
```

Listing A.7: Makefile.in

```
# Makefile.in generated by automake 1.9.6 from Makefile.am.
# @configure_input@
```

```
# Copyright (C) 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
# 2003, 2004, 2005 Free Software Foundation, Inc.
# This Makefile.in is free software; the Free Software Foundation
# gives unlimited permission to copy and/or distribute it,
# with or without modifications, as long as this notice is preserved.

# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY, to the extent permitted by law; without
# even the implied warranty of MERCHANTABILITY or FITNESS FOR A
# PARTICULAR PURPOSE.

@SET_MAKE@

#SUBDIRS = binutils gdb


srcdir = @srcdir@
top_srcdir = @top_srcdir@
VPATH = @srcdir@
pkgdatadir = $(datadir)/@PACKAGE@
pkglibdir = $(libdir)/@PACKAGE@
pkgincludedir = $(includedir)/@PACKAGE@
top_builddir = ../..
am__cd = CDPATH="$${ZSH_VERSION+.}$(PATH_SEPARATOR)" && cd
INSTALL = @INSTALL@
install_sh_DATA = $(install_sh) -c -m 644
install_sh_PROGRAM = $(install_sh) -c
install_sh_SCRIPT = $(install_sh) -c
INSTALL_HEADER = $(INSTALL_DATA)
transform = $(program_transform_name)
NORMAL_INSTALL = :
PRE_INSTALL = :
POST_INSTALL = :
NORMAL_UNINSTALL = :
```

```
PRE_UNINSTALL = :
POST_UNINSTALL = :
build_triplet = @build@
host_triplet = @host@
bin_PROGRAMS = bmdsfg$(EXEEXT)
subdir = src/acbinutils
DIST_COMMON = $(srcdir)/Makefile.am $(srcdir)/Makefile.in
ACLOCAL_M4 = $(top_srcdir)/aclocal.m4
am__aclocal_m4_deps = $(top_srcdir)/configure.ac
am__configure_deps = $(am__aclocal_m4_deps) $(CONFIGURE_DEPENDENCIES) \
        $(ACLOCAL_M4)
mkinstalldirs = $(install_sh) -d
CONFIG_HEADER = $(top_builddir)/config.h
CONFIG_CLEAN_FILES =
am__installdirs = "$(DESTDIR)$(bindir)" "$(DESTDIR)$(bindir)"
binPROGRAMS_INSTALL = $(INSTALL_PROGRAM)
PROGRAMS = $(bin_PROGRAMS)
am_bmdsfg_OBJECTS = main.$(OBJEXT) utils.$(OBJEXT) opcodes.$(OBJEXT) \
        bfd.$(OBJEXT) gas.$(OBJEXT)
bmdsfg_OBJECTS = $(am_bmdsfg_OBJECTS)
bmdsfg_DEPENDENCIES = ../acpp/libacpp.la ../replace/libreplace.la
binSCRIPT_INSTALL = $(INSTALL_SCRIPT)
SCRIPTS = $(bin_SCRIPTS)
DEFAULT_INCLUDES = -I. -I$(srcdir) -I$(top_builddir)
depcomp = $(SHELL) $(top_srcdir)/config/depcomp
am__depfiles_maybe = depfiles
COMPILE = $(CC) $(DEFS) $(DEFAULT_INCLUDES) $(INCLUDES) $(
    AM_CPPFLAGS) \
        $(CPPFLAGS) $(AM_CFLAGS) $(CFLAGS)
LTCOMPILE = $(LIBTOOL) --tag=CC --mode=compile $(CC) $(DEFS) \
        $(DEFAULT_INCLUDES) $(INCLUDES) $(AM_CPPFLAGS) $(CPPFLAGS)
            \
        $(AM_CFLAGS) $(CFLAGS)
CCLD = $(CC)
```

```
LINK = $(LIBTOOL) --tag=CC --mode=link $(CCLD) $(AM_CFLAGS) $(
    CFLAGS) \
        $(AM_LDFLAGS) $(LDFLAGS) -o $@
SOURCES = $(bmdsfg_SOURCES)
DIST_SOURCES = $(bmdsfg_SOURCES)
ETAGS = etags
CTAGS = ctags
DISTFILES = $(DIST_COMMON) $(DIST_SOURCES) $(TEXINFOS) $(
    EXTRA_DIST)
ACLOCAL = @ACLOCAL@
AMDEP_FALSE = @AMDEP_FALSE@
AMDEP_TRUE = @AMDEP_TRUE@
AMTAR = @AMTAR@
AR = @AR@
AUTOCONF = @AUTOCONF@
AUTOHEADER = @AUTOHEADER@
AUTOMAKE = @AUTOMAKE@
AWK = @AWK@
BINUTILS_DIR = @BINUTILS_DIR@
CC = @CC@
CCDEPMODE = @CCDEPMODE@
CFLAGS = @CFLAGS@
CPP = @CPP@
CPPFLAGS = @CPPFLAGS@
CXX = @CXX@
CXXCPP = @CXXCPP@
CXXDEPMODE = @CXXDEPMODE@
CXXFLAGS = @CXXFLAGS@
CYGPATH_W = @CYGPATH_W@
DEFS = @DEFS@
DEPDIR = @DEPDIR@
ECHO = @ECHO@
ECHO_C = @ECHO_C@
ECHO_N = @ECHO_N@
ECHO_T = @ECHO_T@
```

EGREP = @EGREP@
EXEEXT = @EXEEXT@
F77 = @F77@
FFLAGS = @FFLAGS@
GDB_DIR = @GDB_DIR@
INSTALL_DATA = @INSTALL_DATA@
INSTALL_PROGRAM = @INSTALL_PROGRAM@
INSTALL_SCRIPT = @INSTALL_SCRIPT@
INSTALL_STRIP_PROGRAM = @INSTALL_STRIP_PROGRAM@
LDFLAGS = @LDFLAGS@
LEX = @LEX@
LEXLIB = @LEXLIB@
LEX_OUTPUT_ROOT = @LEX_OUTPUT_ROOT@
LIBOBJS = @LIBOBJS@
LIBS = @LIBS@
LIBTOOL = @LIBTOOL@
LN_S = @LN_S@
LTLIBOBJS = @LTLIBOBJS@
MAKEINFO = @MAKEINFO@
OBJEXT = @OBJEXT@
PACKAGE = @PACKAGE@
PACKAGE_BUGREPORT = @PACKAGE_BUGREPORT@
PACKAGE_NAME = @PACKAGE_NAME@
PACKAGE_STRING = @PACKAGE_STRING@
PACKAGE_TARNAME = @PACKAGE_TARNAME@
PACKAGE_VERSION = @PACKAGE_VERSION@
PATH_SEPARATOR = @PATH_SEPARATOR@
RANLIB = @RANLIB@
SC_DIR = @SC_DIR@
SET_MAKE = @SET_MAKE@
SHELL = @SHELL@
STRIP = @STRIP@
SYSTEMC_SUPPORT_FALSE = @SYSTEMC_SUPPORT_FALSE@
SYSTEMC_SUPPORT_TRUE = @SYSTEMC_SUPPORT_TRUE@
TARGET_ARCH = @TARGET_ARCH@

TLM_DIR = @TLM_DIR@
TLM_SUPPORT_FALSE = @TLM_SUPPORT_FALSE@
TLM_SUPPORT_TRUE = @TLM_SUPPORT_TRUE@
VERSION = @VERSION@
YACC = @YACC@
ac_ct_AR = @ac_ct_AR@
ac_ct_CC = @ac_ct_CC@
ac_ct_CXX = @ac_ct_CXX@
ac_ct_F77 = @ac_ct_F77@
ac_ct_RANLIB = @ac_ct_RANLIB@
ac_ct_STRIP = @ac_ct_STRIP@
am__fastdepCC_FALSE = @am__fastdepCC_FALSE@
am__fastdepCC_TRUE = @am__fastdepCC_TRUE@
am__fastdepCXX_FALSE = @am__fastdepCXX_FALSE@
am__fastdepCXX_TRUE = @am__fastdepCXX_TRUE@
am__include = @am__include@
am__leading_dot = @am__leading_dot@
am__quote = @am__quote@
am__tar = @am__tar@
am__untar = @am__untar@
bindir = @bindir@
build = @build@
build_alias = @build_alias@
build_cpu = @build_cpu@
build_os = @build_os@
build_vendor = @build_vendor@
datadir = @datadir@
exec_prefix = @exec_prefix@
host = @host@
host_alias = @host_alias@
host_cpu = @host_cpu@
host_os = @host_os@
host_vendor = @host_vendor@
includedir = @includedir@
infodir = @infodir@

```
install_sh = @install_sh@
libdir = @libdir@
libexecdir = @libexecdir@
localstatedir = @localstatedir@
mandir = @mandir@
mkdir_p = @mkdir_p@
oldincludedir = @oldincludedir@
prefix = @prefix@
program_transform_name = @program_transform_name@
sbindir = @sbindir@
sharedstatedir = @sharedstatedir@
sysconfdir = @sysconfdir@
target_alias = @target_alias@
EXTRA_DIST = acbingen.sh.in \
             binutils \
             gdb

INCLUDES = -I. -I$(top_srcdir)/src/replace -I$(top_srcdir)/src/aclib/ac_decoder -
    I$(top_srcdir)/src/acpp -I$(top_srcdir)/src/acsim
AM_CFLAGS = -DACVERSION=\"$(VERSION)\"
BINUTILS_ROOT = binutils
BINUTILS_TREE = \
        $(BINUTILS_ROOT)/include \
        $(BINUTILS_ROOT)/include/elf \
        $(BINUTILS_ROOT)/include/opcode \
        $(BINUTILS_ROOT)/bfd \
        $(BINUTILS_ROOT)/opcodes \
        $(BINUTILS_ROOT)/gas \
        $(BINUTILS_ROOT)/gas/config \
        $(BINUTILS_ROOT)/ld \
        $(BINUTILS_ROOT)/ld/emulparams

FILES_TO_PATCH = \
        bfd/archures.c \
        bfd/Makefile.in \
```

```
        bfd/bfd-in2.h \
        bfd/config.bfd \
        bfd/configure \
        bfd/targets.c \
        config.sub \
        gas/configure.tgt \
        gas/Makefile.in \
        opcodes/configure \
        opcodes/Makefile.in \
        opcodes/disassemble.c \
        ld/configure.tgt \
        ld/Makefile.in \
        include/dis-asm.h

FILES_TO_COPY = \
        gas/config/tc-xxxxx.c \
        gas/config/tc-xxxxx.h \
        opcodes/xxxxx-opc.c \
        opcodes/xxxxx-dis.c \
        include/opcode/xxxxx.h \
        include/elf/xxxxx.h \
        bfd/elf32-xxxxx.c \
        bfd/cpu-xxxxx.c \
        ld/emulparams/xxxxxelf.sh

GDB_ROOT = gdb
GDB_TREE = \
        $(GDB_ROOT)/include \
        $(GDB_ROOT)/include/elf \
        $(GDB_ROOT)/include/opcode \
        $(GDB_ROOT)/bfd \
        $(GDB_ROOT)/opcodes \
        $(GDB_ROOT)/gdb \
        $(GDB_ROOT)/gdb/config/xxxxx
```

```
FILES_TO_PATCH_GDB = \
        bfd/archures.c \
        bfd/Makefile.in \
        bfd/bfd-in2.h \
        bfd/config.bfd \
        bfd/configure \
        bfd/targets.c \
        config.sub \
        gdb/configure.host \
        gdb/configure.tgt \
        gdb/Makefile.in \
        opcodes/configure \
        opcodes/Makefile.in \
        opcodes/disassemble.c \
        include/dis-asm.h

FILES_TO_COPY_GDB = \
        gdb/xxxxx-tdep.c \
        gdb/config/xxxxx/xxxxx.mt \
        opcodes/xxxxx-opc.c \
        opcodes/xxxxx-dis.c \
        include/opcode/xxxxx.h \
        include/elf/xxxxx.h \
        bfd/elf32-xxxxx.c \
        bfd/cpu-xxxxx.c

bmdsfg_SOURCES = main.c utils.c utils.h opcodes.c opcodes.h bfd.c bfd.h gas.c gas.h
bmdsfg_LDADD = ../acpp/libacpp.la ../replace/libreplace.la
edit = sed \
        -e 's,@pkgdatadir\@,$(pkgdatadir),g' \
        -e 's,@sysconfdir\@,$(sysconfdir),g' \
        -e 's,@bindir\@,${bindir},g' \
        -e 's,@prefix\@,$(prefix),g' \
        -e 's,@patchfiles\@,$(FILES_TO_PATCH),g' \
        -e 's,@copyfiles\@,$(FILES_TO_COPY),g' \
```

```
        -e 's,@patchfilesgdb\@,$(FILES_TO_PATCH_GDB),g' \
        -e 's,@copyfilesgdb\@,$(FILES_TO_COPY_GDB),g' \
        -e 's,@VERSION\@,$(VERSION),g'

bin_SCRIPTS = acbingen.sh
all: all-am

.SUFFIXES:
.SUFFIXES: .c .lo .o .obj
$(srcdir)/Makefile.in: $(srcdir)/Makefile.am $(am__configure_deps)
        @for dep in $?; do \
          case '$(am__configure_deps)' in \
            *$$dep*) \
              cd $(top_builddir) && $(MAKE) $(AM_MAKEFLAGS) am--refresh \
                && exit 0; \
              exit 1;; \
          esac; \
        done; \
        echo ' cd $(top_srcdir) && $(AUTOMAKE) --foreign src/acbinutils/Makefile
            '; \
        cd $(top_srcdir) && \
          $(AUTOMAKE) --foreign src/acbinutils/Makefile
.PRECIOUS: Makefile
Makefile: $(srcdir)/Makefile.in $(top_builddir)/config.status
        @case '$?' in \
          *config.status*) \
            cd $(top_builddir) && $(MAKE) $(AM_MAKEFLAGS) am--refresh;; \
          *) \
            echo ' cd $(top_builddir) && $(SHELL) ./config.status $(subdir)/$@ $(
                am__depfiles_maybe)'; \
            cd $(top_builddir) && $(SHELL) ./config.status $(subdir)/$@ $(
                am__depfiles_maybe);; \
        esac;
```

```
$(top_builddir)/config.status: $(top_srcdir)/configure $(
    CONFIG_STATUS_DEPENDENCIES)
        cd $(top_builddir) && $(MAKE) $(AM_MAKEFLAGS) am--refresh

$(top_srcdir)/configure: $(am__configure_deps)
        cd $(top_builddir) && $(MAKE) $(AM_MAKEFLAGS) am--refresh
$(ACLOCAL_M4): $(am__aclocal_m4_deps)
        cd $(top_builddir) && $(MAKE) $(AM_MAKEFLAGS) am--refresh
install-binPROGRAMS: $(bin_PROGRAMS)
        @$(NORMAL_INSTALL)
        test -z "$(bindir)" || $(mkdir_p) "$(DESTDIR)$(bindir)"
        @list='$(bin_PROGRAMS)'; for p in $$list; do \
          p1=`echo $$p|sed 's/$(EXEEXT)$$//'`; \
          if test -f $$p \
             || test -f $$p1 \
          ; then \
            f=`echo "$$p1" | sed 's,^.*/,,;$(transform);s/$$/$(EXEEXT)/'`; \
           echo " $(INSTALL_PROGRAM_ENV) $(LIBTOOL) --mode=install $(
               binPROGRAMS_INSTALL) '$$p' '$(DESTDIR)$(bindir)/$$f'"; \
           $(INSTALL_PROGRAM_ENV) $(LIBTOOL) --mode=install $(
               binPROGRAMS_INSTALL) "$$p" "$(DESTDIR)$(bindir)/$$f" || exit
               1; \
          else :; fi; \
        done

uninstall-binPROGRAMS:
        @$(NORMAL_UNINSTALL)
        @list='$(bin_PROGRAMS)'; for p in $$list; do \
          f=`echo "$$p" | sed 's,^.*/,,;s/$(EXEEXT)$$//;$(transform);s/$$/$(
              EXEEXT)/'`; \
          echo " rm -f '$(DESTDIR)$(bindir)/$$f'"; \
          rm -f "$(DESTDIR)$(bindir)/$$f"; \
        done

clean-binPROGRAMS:
```

```
        @list='$(bin_PROGRAMS)'; for p in $$list; do \
          f=`echo $$p|sed 's/$(EXEEXT)$$//'`; \
          echo " rm -f $$p $$f"; \
          rm -f $$p $$f ; \
        done
bmdsfg$(EXEEXT): $(bmdsfg_OBJECTS) $(bmdsfg_DEPENDENCIES)
        @rm -f bmdsfg$(EXEEXT)
        $(LINK) $(bmdsfg_LDFLAGS) $(bmdsfg_OBJECTS) $(bmdsfg_LDADD) $(
            LIBS)
install-binSCRIPTS: $(bin_SCRIPTS)
        @$(NORMAL_INSTALL)
        test -z "$(bindir)" || $(mkdir_p) "$(DESTDIR)$(bindir)"
        @list='$(bin_SCRIPTS)'; for p in $$list; do \
          if test -f "$$p"; then d=; else d="$(srcdir)/"; fi; \
          if test -f $$d$$p; then \
            f=`echo "$$p" | sed 's|^.*/||;$(transform)'`; \
            echo " $(binSCRIPT_INSTALL) '$$d$$p' '$(DESTDIR)$(bindir)/$$f'"; \
            $(binSCRIPT_INSTALL) "$$d$$p" "$(DESTDIR)$(bindir)/$$f"; \
          else :; fi; \
        done

uninstall-binSCRIPTS:
        @$(NORMAL_UNINSTALL)
        @list='$(bin_SCRIPTS)'; for p in $$list; do \
          f=`echo "$$p" | sed 's|^.*/||;$(transform)'`; \
          echo " rm -f '$(DESTDIR)$(bindir)/$$f'"; \
          rm -f "$(DESTDIR)$(bindir)/$$f"; \
        done

mostlyclean-compile:
        -rm -f *.$(OBJEXT)

distclean-compile:
        -rm -f *.tab.c
```

```
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/bfd.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/gas.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/main.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/opcodes.
    Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/utils.Po@am__quote@

.c.o:
@am__fastdepCC_TRUE@ if $(COMPILE) -MT $@ -MD -MP -MF "$(DEPDIR)/$
    *.Tpo" -c -o $@ $<; \
@am__fastdepCC_TRUE@ then mv -f "$(DEPDIR)/$*.Tpo" "$(DEPDIR)/$*.Po"; else
     rm -f "$(DEPDIR)/$*.Tpo"; exit 1; fi
@AMDEP_TRUE@@am__fastdepCC_FALSE@ source='$<' object='$@' libtool=no
    @AMDEPBACKSLASH@
@AMDEP_TRUE@@am__fastdepCC_FALSE@ DEPDIR=$(DEPDIR) $(CCDEPMODE
    ) $(depcomp) @AMDEPBACKSLASH@
@am__fastdepCC_FALSE@ $(COMPILE) -c $<

.c.obj:
@am__fastdepCC_TRUE@ if $(COMPILE) -MT $@ -MD -MP -MF "$(DEPDIR)/$
    *.Tpo" -c -o $@ `$(CYGPATH_W) '$<'`; \
@am__fastdepCC_TRUE@ then mv -f "$(DEPDIR)/$*.Tpo" "$(DEPDIR)/$*.Po"; else
     rm -f "$(DEPDIR)/$*.Tpo"; exit 1; fi
@AMDEP_TRUE@@am__fastdepCC_FALSE@ source='$<' object='$@' libtool=no
    @AMDEPBACKSLASH@
@AMDEP_TRUE@@am__fastdepCC_FALSE@ DEPDIR=$(DEPDIR) $(CCDEPMODE
    ) $(depcomp) @AMDEPBACKSLASH@
@am__fastdepCC_FALSE@ $(COMPILE) -c `$(CYGPATH_W) '$<'`

.c.lo:
@am__fastdepCC_TRUE@ if $(LTCOMPILE) -MT $@ -MD -MP -MF "$(DEPDIR)
    /$*.Tpo" -c -o $@ $<; \
@am__fastdepCC_TRUE@ then mv -f "$(DEPDIR)/$*.Tpo" "$(DEPDIR)/$*.Plo";
    else rm -f "$(DEPDIR)/$*.Tpo"; exit 1; fi
```

```
@AMDEP_TRUE@@am__fastdepCC_FALSE@ source='$<' object='$@' libtool=yes
    @AMDEPBACKSLASH@
@AMDEP_TRUE@@am__fastdepCC_FALSE@ DEPDIR=$(DEPDIR) $(CCDEPMODE
    ) $(depcomp) @AMDEPBACKSLASH@
@am__fastdepCC_FALSE@ $(LTCOMPILE) -c -o $@ $<

mostlyclean-libtool:
        -rm -f *.lo

clean-libtool:
        -rm -rf .libs _libs

distclean-libtool:
        -rm -f libtool
uninstall-info-am:

ID: $(HEADERS) $(SOURCES) $(LISP) $(TAGS_FILES)
        list='$(SOURCES) $(HEADERS) $(LISP) $(TAGS_FILES)'; \
        unique=`for i in $$list; do \
            if test -f "$$i"; then echo $$i; else echo $(srcdir)/$$i; fi; \
          done | \
          $(AWK) ' { files[$$0] = 1; } \
               END { for (i in files) print i; }'`; \
        mkid -fID $$unique
tags: TAGS

TAGS: $(HEADERS) $(SOURCES) $(TAGS_DEPENDENCIES) \
                $(TAGS_FILES) $(LISP)
        tags=; \
        here=`pwd`; \
        list='$(SOURCES) $(HEADERS) $(LISP) $(TAGS_FILES)'; \
        unique=`for i in $$list; do \
            if test -f "$$i"; then echo $$i; else echo $(srcdir)/$$i; fi; \
          done | \
          $(AWK) ' { files[$$0] = 1; } \
```

```
            END { for (i in files) print i; }'`; \
        if test -z "$(ETAGS_ARGS)$$tags$$unique"; then :; else \
          test -n "$$unique" || unique=$$empty_fix; \
          $(ETAGS) $(ETAGSFLAGS) $(AM_ETAGSFLAGS) $(ETAGS_ARGS) \
            $$tags $$unique; \
        fi
ctags: CTAGS
CTAGS: $(HEADERS) $(SOURCES) $(TAGS_DEPENDENCIES) \
             $(TAGS_FILES) $(LISP)
        tags=; \
        here=`pwd`; \
        list='$(SOURCES) $(HEADERS) $(LISP) $(TAGS_FILES)'; \
        unique=`for i in $$list; do \
            if test -f "$$i"; then echo $$i; else echo $(srcdir)/$$i; fi; \
          done | \
          $(AWK) ' { files[$$0] = 1; } \
               END { for (i in files) print i; }'`; \
        test -z "$(CTAGS_ARGS)$$tags$$unique" \
          || $(CTAGS) $(CTAGSFLAGS) $(AM_CTAGSFLAGS) $(CTAGS_ARGS) \
             $$tags $$unique

GTAGS:
        here=`$(am__cd) $(top_builddir) && pwd` \
          && cd $(top_srcdir) \
          && gtags -i $(GTAGS_ARGS) $$here

distclean-tags:
        -rm -f TAGS ID GTAGS GRTAGS GSYMS GPATH tags

distdir: $(DISTFILES)
        @srcdirstrip=`echo "$(srcdir)" | sed 's|.|.|g'`; \
        topsrcdirstrip=`echo "$(top_srcdir)" | sed 's|.|.|g'`; \
        list='$(DISTFILES)'; for file in $$list; do \
          case $$file in \
            $(srcdir)/*) file=`echo "$$file" | sed "s|^$$srcdirstrip/||"`;; \
```

```
      $(top_srcdir)/*) file=`echo "$$file" | sed "s|^$$topsrcdirstrip/|$(top_builddir
          )/|"`;; \
    esac; \
    if test -f $$file || test -d $$file; then d=.; else d=$(srcdir); fi; \
    dir=`echo "$$file" | sed -e 's,/[^/]*$$,,'`; \
    if test "$$dir" != "$$file" && test "$$dir" != "."; then \
      dir="/$$dir"; \
      $(mkdir_p) "$(distdir)$$dir"; \
    else \
      dir=''; \
    fi; \
    if test -d $$d/$$file; then \
      if test -d $(srcdir)/$$file && test $$d != $(srcdir); then \
        cp -pR $(srcdir)/$$file $(distdir)$$dir || exit 1; \
      fi; \
      cp -pR $$d/$$file $(distdir)$$dir || exit 1; \
    else \
      test -f $(distdir)/$$file \
      || cp -p $$d/$$file $(distdir)/$$file \
      || exit 1; \
    fi; \
  done
check-am: all-am
check: check-am
all-am: Makefile $(PROGRAMS) $(SCRIPTS)
installdirs:
  for dir in "$(DESTDIR)$(bindir)" "$(DESTDIR)$(bindir)"; do \
    test -z "$$dir" || $(mkdir_p) "$$dir"; \
  done
install: install-am
install-exec: install-exec-am
install-data: install-data-am
uninstall: uninstall-am

install-am: all-am
```

```
	@$(MAKE) $(AM_MAKEFLAGS) install-exec-am install-data-am

installcheck: installcheck-am
install-strip:
	$(MAKE) $(AM_MAKEFLAGS) INSTALL_PROGRAM="$(
	    INSTALL_STRIP_PROGRAM)" \
	  install_sh_PROGRAM="$(INSTALL_STRIP_PROGRAM)"
	      INSTALL_STRIP_FLAG=-s \
	  `test -z '$(STRIP)' || \
	    echo "INSTALL_PROGRAM_ENV=STRIPPROG='$(STRIP)'"` install
mostlyclean-generic:

clean-generic:

distclean-generic:
	-test -z "$(CONFIG_CLEAN_FILES)" || rm -f $(CONFIG_CLEAN_FILES)

maintainer-clean-generic:
	@echo "This command is intended for maintainers to use"
	@echo "it deletes files that may require special tools to rebuild."
clean: clean-am

clean-am: clean-binPROGRAMS clean-generic clean-libtool mostlyclean-am

distclean: distclean-am
	-rm -rf ./$(DEPDIR)
	-rm -f Makefile
distclean-am: clean-am distclean-compile distclean-generic \
	distclean-libtool distclean-tags

dvi: dvi-am

dvi-am:

html: html-am
```

```
info: info-am

info-am:

install-data-am:
	@$(NORMAL_INSTALL)
	$(MAKE) $(AM_MAKEFLAGS) install-data-hook

install-exec-am: install-binPROGRAMS install-binSCRIPTS

install-info: install-info-am

install-man:

installcheck-am:

maintainer-clean: maintainer-clean-am
	-rm -rf ./$(DEPDIR)
	-rm -f Makefile
maintainer-clean-am: distclean-am maintainer-clean-generic

mostlyclean: mostlyclean-am

mostlyclean-am: mostlyclean-compile mostlyclean-generic \
	mostlyclean-libtool

pdf: pdf-am

pdf-am:

ps: ps-am

ps-am:
```

```
uninstall-am: uninstall-binPROGRAMS uninstall-binSCRIPTS \
        uninstall-info-am
        @$(NORMAL_INSTALL)
        $(MAKE) $(AM_MAKEFLAGS) uninstall-hook

.PHONY: CTAGS GTAGS all all-am check check-am clean clean-binPROGRAMS \
        clean-generic clean-libtool ctags distclean distclean-compile \
        distclean-generic distclean-libtool distclean-tags distdir dvi \
        dvi-am html html-am info info-am install install-am \
        install-binPROGRAMS install-binSCRIPTS install-data \
        install-data-am install-data-hook install-exec install-exec-am \
        install-info install-info-am install-man install-strip \
        installcheck installcheck-am installdirs maintainer-clean \
        maintainer-clean-generic mostlyclean mostlyclean-compile \
        mostlyclean-generic mostlyclean-libtool pdf pdf-am ps ps-am \
        tags uninstall uninstall-am uninstall-binPROGRAMS \
        uninstall-binSCRIPTS uninstall-hook uninstall-info-am


acbingen.sh: Makefile $(srcdir)/acbingen.sh.in
        $(edit) $(srcdir)/acbingen.sh.in > acbingen.sh

install-data-hook:
        for directory in $(BINUTILS_ROOT) $(BINUTILS_TREE); do \
          mkdir -p $(pkgdatadir)/$${directory}; \
        done;
        for file in $(FILES_TO_PATCH); do \
          cp -f $(srcdir)/$(BINUTILS_ROOT)/$${file}.sed $(pkgdatadir)/$(
              BINUTILS_ROOT)/$${file}.sed; \
        done;
        for file in $(FILES_TO_COPY); do \
          cp -f $(srcdir)/$(BINUTILS_ROOT)/$${file} $(pkgdatadir)/$(
              BINUTILS_ROOT)/$${file}; \
        done
        for directory in $(GDB_ROOT) $(GDB_TREE); do \
```

```
        mkdir -p $(pkgdatadir)/$${directory}; \
      done;
      for file in $(FILES_TO_PATCH_GDB); do \
        cp -f $(srcdir)/$(GDB_ROOT)/$${file}.sed $(pkgdatadir)/$(GDB_ROOT)/
            $${file}.sed; \
      done;
      for file in $(FILES_TO_COPY_GDB); do \
        cp -f $(srcdir)/$(GDB_ROOT)/$${file} $(pkgdatadir)/$(GDB_ROOT)/$${
            file}; \
      done

uninstall-hook:
      rm -rf $(pkgdatadir)/$(BINUTILS_ROOT)
      rm -rf $(pkgdatadir)/$(GDB_ROOT)
      rm acbingen.sh
# Tell versions [3.59,3.63) of GNU make to not export all variables.
# Otherwise a system limit (for SysV at least) may be exceeded.
.NOEXPORT:
```

Listing A.8: opcodes.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file opcodes.c
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC-UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
```

```
 * System Design Automation Lab (LAPS)
 * INE-UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 -0300
 *
 * @brief Opcodes library related code (implementation)
 *
 * @attention Copyright (C) 2002-2006 --- The ArchC Team
 *
 */


#include <stdlib.h>
#include <stdio.h>

#include "utils.h"
#include "opcodes.h"

/*
 * module function prototypes
 */
static void create_operand_string(ac_asm_insn *insn, char **output);
static unsigned int encode_insn_field(unsigned int field_value, unsigned insn_size,
    unsigned fbit, unsigned fsize);
static unsigned int encode_dmask_field(unsigned insn_size, unsigned fbit, unsigned
     fsize);



/*
   Opcode table generation

  This table is of type acasm_opcode with the following structure:
```

```
. const char *mnemonic
   The mnemonic string (cannot be NULL)

. const char *args
   A string with the syntax of the operands (may be NULL)
   The following format is used:
     args ::= [literal_chars] (operand_tag literal_chars)*
     operand_tag ::= '%' format_specifier ':' field_id ':'

     format_specifier: may be a user defined or a built-in specifier (a string)
     literal_chars: a sequence of valid chars which the assembler needs to match
         literally
     field_id: an integer representing the field of this operand in the instruction format

. unsigned long image
   The base binary image of this instruction

. unsigned long format_id
   The format identifier. 0 is the first format in the list, 1 is the second, and so on

. unsigned long pseudo_idx
   An index to the first instruction to be executed in case this is a pseudo instruction

. unsigned long counter
   (optional) Used to count the number of instruction assembled

*/
int CreateOpcodeTable(const char *table_filename)
{
  char *strP;
  FILE *output;

  if ((output = fopen(table_filename, "w")) == NULL)
    return 0;
```

```
/* get the instruction list from parser */
ac_asm_insn *asml = ac_asm_get_asm_insn_list();

long pseudo_idx = 1; /* index to an instruction in the pseudo_instr table */
while (asml != NULL) {

  /* mnemonic */
  /*
     ----------------------------------------------------------------
     */
  fprintf(output, "%s{\"%s\",\t", IND1, asml->mnemonic);
  /*
     ----------------------------------------------------------------
     */

  /* args string */
  /*
     ----------------------------------------------------------------
     */
  char *buffer = (char *)malloc(100);
  create_operand_string(asml, &buffer);
  fprintf(output, "\"%s\",\t", buffer);
  free(buffer);
  /*
     ----------------------------------------------------------------
     */

  /* base image */
  /*
     ----------------------------------------------------------------
     */
  unsigned long base_image = 0x00;
  unsigned long dmask = 0x00;
  unsigned long format_id = 99; /* a pseudo_instr has a 99 format id */
```

```
ac_dec_format *pfrm = format_ins_list;

if (asml->insn != NULL) { /* native instructions */

  /* get the format of this instruction. 0 is the first */
  format_id = 0;
  while ((pfrm != NULL) && strcmp(asml->insn->format, pfrm->name)) {
    format_id++;
    pfrm = pfrm->next;
  }

  if (pfrm == NULL) internal_error();
  else {
    /* for each decoding field, finds its place in the format and encode it in
    the base_image variable */
      {
        ac_dec_list *pdl = asml->insn->dec_list;

        while (pdl != NULL) {
          ac_dec_field *pdf = pfrm->fields;
          while ((pdf != NULL) && strcmp(pdl->name, pdf->name)) pdf = pdf
              ->next;

          if (pdf == NULL) internal_error();
          else {
            base_image |= encode_insn_field(pdl->value, pfrm->size, pdf->
                first_bit, pdf->size);
            dmask |= encode_dmask_field(pfrm->size, pdf->first_bit, pdf->size);
          }

          pdl = pdl->next;
        }

        ac_const_field_list *cfP = asml->const_fields;
```

```
          while (cfP != NULL) {
            base_image |= encode_insn_field(cfP->value, pfrm->size, cfP->field.
                first_bit, cfP->field.size);
            dmask |= encode_dmask_field(pfrm->size, cfP->field.first_bit, cfP->
                field.size);
            cfP = cfP->next;
          }
        }

    }
}
fprintf(output, "0x%08X,\t", base_image);
/*
    ------------------------------------------------------------
    */

/* format id */
/*
    ------------------------------------------------------------
    */
fprintf(output, "%d,\t", format_id);
/*
    ------------------------------------------------------------
    */

/* if a pseudo, saves the pseudo-instr table instruction index */
/*
    ------------------------------------------------------------
    */
if (asml->pseudolist != NULL) {
  fprintf(output, "%d", pseudo_idx);
  pseudo_idx += asml->num_pseudo+1;
}
else
  fprintf(output, "0");
```

```
    /*
        ------------------------------------------------------------
        */

    /***
     Optional statistic field
     ***/
    fprintf(output, ",\t0");

    /* Mask, used for disassemble */
    fprintf(output, ", \t0x%08X", dmask);

    fprintf(output, "},\n");

    asml = asml->next;
  } /* end-of-while */

  fclose(output);
  return 1;
}

/*
  User defined symbol table generation

  TODO: describe the acasm_symbol structure

*/
int CreateAsmSymbolTable(const char *symtab_filename)
{
  FILE *output;

  if ((output = fopen(symtab_filename, "w")) == NULL)
    return 0;

  /* gets the mappings */
```

```
  ac_asm_map_list *ml = ac_asm_get_mapping_list();

  /* for each conversion specifier... */
  while (ml != NULL) {

    if (!(ml->used_where & 1)) { /* skip if not an operand marker */
      ml = ml->next;
      continue;
    }

    ac_asm_symbol *s = ml->symbol_list;

    /* ... and for each symbol, write the mapping */
    while (s != NULL) {
      fprintf(output, "%s{\"%s\",\t\"%s\",\t%d},\n", IND1, s->symbol, ml->
          marker,
            s->value);

      s = s->next;
    }

    ml = ml->next;
  }

  fclose(output);
  return 1;
}



/*
  Pseudo instruction table generation

  TODO: describe the structure

*/
```

```
int CreatePseudoOpsTable(const char *optable_filename)
{
  FILE *output;

  if ((output = fopen(optable_filename, "w")) == NULL)
    return 0;

  /* gets the list from the parser */
  ac_asm_insn *asml = ac_asm_get_asm_insn_list();

  /* first entry is always NULL */
  fprintf(output, "%sNULL", IND1);

  /* for each insn... */
  while (asml != NULL) {

    /* ... if it's a pseudo-instr, write the list of instructions in sequence */
    if (asml->pseudolist != NULL) {

      strlist *pl = asml->pseudolist;
      while (pl != NULL) {
        fprintf(output, ",\n%s\"%s\"", IND1, pl->str);
        pl = pl->next;
      }

      fprintf(output, ",\n%sNULL", IND1);
    }

    asml = asml->next;
  }

  fclose(output);
  return 1;
}
```

```
/*
 * module function implementations
 */

static void create_operand_string(ac_asm_insn *insn, char **output)
{
  char *s = *output;
  char *lit = insn->op_literal;

  *s = '\0';

  ac_operand_list *opP = insn->operands;

  while (*lit != '\0') {

    if (*lit == '%') {
      *s = *lit;
      s++;
      lit++;

      char *ls = opP->str;
      while (*ls != '\0') {
        *s = *ls;
        s++;
        ls++;
      }

      *s = ':';

      ac_asm_insn_field *fP = opP->fields;
      while (fP != NULL) {
```

```
      s++;
      sprintf(s, "%d", fP->id);
      while (*s >= '0' && *s <= '9') s++;

      *s = ':';

      s++;
      sprintf(s, "%d", fP->reloc_id);
      while (*s >= '0' && *s <= '9') s++;

      fP = fP->next;
      if (fP != NULL)
        *s = '+';
    }

    *s = ':';
    s++;

    opP = opP->next;
    /* TODO: check for NULL pointer */

  }
  else if (*lit == '\\') {
    *s = *lit;
    s++;
    lit++;
    if (*lit == '\\') {
      *s = *lit;
      s++;
      lit++;

      if (*lit == '%') {
        *s = *lit;
        s++;
        lit++;
```

```
        }
      }
    }
    else {
      *s = *lit;
      s++;
      lit++;
    }
  }
  *s = '\0';
}

static unsigned int encode_insn_field(unsigned int field_value, unsigned insn_size,
    unsigned fbit, unsigned fsize) {

  // TODO: see if the 'val' field can accept constant values (this is not being checked
      atm)
  // TODO: check if it is correct for different word size instructions
  unsigned int mask1 = 0xffffffff;
  unsigned int mask2 = 0xffffffff;
  unsigned int return_value;

  mask1 <<= (insn_size-(fbit+1));
  mask2 >>= fbit+1-fsize;
  return_value = ((field_value << (insn_size-(fbit+1)) & (mask1 & mask2)));

  return return_value;
}

static unsigned int encode_dmask_field(unsigned insn_size, unsigned fbit, unsigned
    fsize)
{
  // TODO: see if the 'val' field can accept constant values (this is not being checked
      atm)
  unsigned int mask1 = 0xffffffff;
```

```
  unsigned int mask2 = 0xffffffff;
  unsigned int field_value=0;

  int i=0;
  for (i;i<fsize;i++){
    field_value = (field_value<<1) + 1;
  }

  mask1 <<= (insn_size-(fbit+1));
  mask2 >>= fbit+1-fsize;
  return ((field_value << (insn_size-(fbit+1)) & (mask1 & mask2)));
}
```

Listing A.9: opcodes.h

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file opcodes.h
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC-UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE-UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 -0300
```

```
 *
 * @brief Opcodes library related code
 *
 * The Opcodes module deals with the aspects involved in retargeting
 * the Opcodes library.
 *
 * @attention Copyright (C) 2002--2006 --- The ArchC Team
 *
 */

/** @defgroup opcodes_group Opcodes library
 * @ingroup binutils_group
 *
 * @{
 */

#ifndef _OPCODES_H_
#define _OPCODES_H_

extern int CreateOpcodeTable(const char *table_filename);
extern int CreateAsmSymbolTable(const char *symtab_filename);
extern int CreatePseudoOpsTable(const char *optable_filename);

#endif /* _OPCODES_H_ */

/* @} */
```

Listing A.10: utils.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file utils.c
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
```

```
 *
 * @author The ArchC Team
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC−UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE−UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 −0300
 *
 * @brief Utilities routines for the acbinutils module
 * (implementation)
 *
 * @attention Copyright (C) 2002−2006 −−− The ArchC Team
 *
 */

#include <stdlib.h>
#include "utils.h"


static char *arch_name = NULL; /* name of the architecture */

/* assuming max input = 64 */
int log_table[] = { 0 /*invalid*/, 0, 1, 1,
                    2 /* log 4 */, 2, 2, 2,
                    3 /* log 8 */, 3, 3, 3,
                    3, 3, 3, 3,
                    4 /* log 16 */, 4, 4, 4 };
```

```
/* General internal error handling function */
void internal_error() {
  printf("Internal Error. Contact the ArchC Team.\n");
  exit(-1);
}

/*
 * Returns the size of the architecture word (in bits)
 * It uses the ac_wordsize currently
 * Note that it only support architectures with, at most, 32-bit words
 */
unsigned int get_arch_size()
{
  extern int wordsize;
  return wordsize ? wordsize : 32;
}

/*
 * Returns the size of an instruction format (in bits)
 */
unsigned int get_insn_size(ac_asm_insn *insn)
{
  extern ac_dec_format *format_ins_list;
  ac_dec_format *pfrm = format_ins_list;

  if (insn->insn != NULL) {

    while ((pfrm != NULL) && strcmp(insn->insn->format, pfrm->name))
        pfrm = pfrm->next;

    if (pfrm == NULL) internal_error();
    else return pfrm->size;
  }
  else internal_error();
}
```

```
void set_arch_name(char *str)
{
  if (arch_name != NULL) {
    free(arch_name);
    arch_name = NULL;
  }

  if (str == NULL)
    return;

  arch_name = (char *) malloc(strlen(str)+1);
  strcpy(arch_name, str);
}

char *get_arch_name()
{
  return arch_name;
}

/*
 * Returns the max format size of the architecture word (in bits)
 */
unsigned int get_max_format_size()
{
  extern ac_dec_format *format_ins_list;
  ac_dec_format *pfrm = format_ins_list;
  int max_size = 0;

  while (pfrm != NULL) {
    if (pfrm->size > max_size)
      max_size = pfrm->size;
    pfrm = pfrm->next;
  }
```

```
  return max_size;
}

/*
 * Returns if the architecture has variable formats (1) or no (0)
 */
unsigned int get_variable_format_size()
{
  extern ac_dec_format *format_ins_list;
  ac_dec_format *pfrm = format_ins_list;

  int variable_format_size = 0;
  int size = pfrm->size;

  while (pfrm != NULL) {
    if (pfrm->size != size) {
      variable_format_size = 1;
      break;
    }
    pfrm = pfrm->next;
  }

  return variable_format_size;
}
```

Listing A.11: utils.h

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/**
 * @file utils.h
 * @author Alexandro Baldassin (UNICAMP)
 * Daniel Casarotto (UFSC)
 * Max Schultz (UFSC)
 *
 * @author The ArchC Team
```

```
 * http://www.archc.org/
 *
 * Computer Systems Laboratory (LSC)
 * IC−UNICAMP
 * http://www.lsc.ic.unicamp.br/
 *
 * System Design Automation Lab (LAPS)
 * INE−UFSC
 * http://www.laps.inf.ufsc.br/
 *
 * @version 1.0
 * @date Thu, 01 Jun 2006 14:28:06 −0300
 *
 * @brief Utilities routines for the acbinutils module
 *
 * This module implements common routines shared by the
 * binary utilities generation files.
 *
 * @attention Copyright (C) 2002−2006 −−− The ArchC Team
 *
 */

/** @defgroup utils_group Utilities routines
 * @ingroup binutils_group gas_group bfd_group opcodes_group
 *
 * @{
 */

#ifndef _UTILS_H_
#define _UTILS_H_

#include "acpp.h"

/* Indentation */
#define IND1 "  "
```

```
#define IND2 "␣␣␣␣"
#define IND3 "␣␣␣␣␣␣"
#define IND4 "␣␣␣␣␣␣␣␣"
#define IND5 "␣␣␣␣␣␣␣␣␣␣"

extern unsigned int get_arch_size();
extern unsigned int get_insn_size(ac_asm_insn *insn);
extern unsigned int get_max_format_size();
extern unsigned int get_variable_format_size();
extern void set_arch_name(char *str);
extern char *get_arch_name();
extern void internal_error();

extern int log_table[];

/* got from architecture file by the parser */
extern char *isa_filename;
extern int ac_tgt_endian; /* filled by the parser 0=l, 1=b */

extern ac_dec_format *format_ins_list;



#endif /* _UTILS_H_ */

/* @} */
```

Listing A.12: binutils/config.sub.sed

```
/1750a | 580 \\/ a\
        | xxxxx \\

/580-\* \\/ a\
        | xxxxx-* \\

/\*-acorn)/ i\
        xxxxx-*) \
                os=-elf \
```

```
    ;;
```

Listing A.13: binutils/bfd/archures.c.sed

```
/extern const bfd_arch_info_type bfd_a29k_arch;/ i\
extern const bfd_arch_info_type bfd_xxxxx_arch;
```

Listing A.14: binutils/bfd/bfd-in2.h.sed

```
/bfd_arch_last/ i\
  bfd_arch_xxxxx,
```

Listing A.15: binutils/bfd/config.bfd.sed

```
/alpha\*)/ i\
xxxxx)  targ_archs=bfd_xxxxx_arch ;;

/am33_2.0-\*-linux\*)/ i\
  xxxxx-*-elf) \
    targ_defvec=bfd_elf32_xxxxx_vec \
    ;;
```

Listing A.16: binutils/bfd/configure.sed

```
/a29kcoff_big_vec)/ i\
    bfd_elf32_xxxxx_vec) tb="$tb elf32-xxxxx.lo elf32.lo $elf" ;;
```

Listing A.17: binutils/bfd/cpu-xxxxx.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* bfd back-end for ___arch_name___ support
   Copyright 2005, 2006 --- The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of BFD, the Binary File Descriptor library.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
```

(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
*Foundation, Inc., 59 Temple Place − Suite 330, Boston, MA 02111−1307, USA. */*

```
/*
 * written by:
 * Alexandro Baldassin
 */

#include "bfd.h"
#include "sysdep.h"
#include "libbfd.h"

const 'bfd_arch_info_type bfd_'___arch_name___'_arch' = {
  ___word_size___, /* bits in a word. */
  32, /* bits in an address. */
  8, /* bits in a byte. */
  'bfd_arch_'___arch_name___, /* enum bfd_architecture arch. */
  0, /* machine number */
  '"'___arch_name___'"', /* arch_name. */
  '"'___arch_name___'"', /* printable name. */
  3, /* unsigned int section alignment power. */
  TRUE, /* the one and only. */
  bfd_default_compatible,
  bfd_default_scan,
  NULL
};
```

Listing A.18: binutils/bfd/elf32-xxxxx.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* ___arch_name___'-specific' support for ELF
   Copyright 2005, 2006 --- The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of BFD, the Binary File Descriptor library.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA. */

/*
 * written by:
 * Alexandro Baldassin
 * Daniel Casarotto
 */

#include "bfd.h"
#include "sysdep.h"
#include "libbfd.h"
#include "elf-bfd.h"
#include '"elf/'___arch_name___'.h"'
```

```
#include "libiberty.h"

/*
 BFD prior to version 2.16 does not include bfd_get_section_limit
*/
#ifndef bfd_get_section_limit
#define bfd_get_section_limit(bfd, sec) \
  (((sec)->_cooked_size) \
   / bfd_octets_per_byte (bfd))
#endif

static reloc_howto_type* ___arch_name___'_reloc_type_lookup'(bfd_*,
    bfd_reloc_code_real_type);
static_void____arch_name___'_elf_info_to_howto' (bfd *, arelent *, Elf_Internal_Rela *);
static long long getbits(unsigned int bitsize, char *location, int endian);
static void putbits(unsigned int bitsize, char *location, long long value, int endian)
    ;


/* ArchC generic relocation routine prototype*/
static bfd_reloc_status_type
bfd_elf_archc_reloc (bfd *abfd,
                     arelent *reloc_entry,
                     asymbol *symbol,
                     void *data,
                     asection *input_section,
                     bfd *output_bfd,
                     char **error_message ATTRIBUTE_UNUSED);


/* carry relocation function prototype */
bfd_reloc_status_type
'_bfd_'___arch_name___'_elf_carry_reloc' (bfd *abfd, arelent *reloc_entry, asymbol *symbol
    ,
```

```
            void *data, asection *input_section, bfd *output_bfd, char **error_message)
                ;

/* Local label (those who starts with $ are recognized as well) */
static bfd_boolean
___arch_name___'_elf_is_local_label_name' (bfd *abfd, const char *name)
{
  if (name[0] == '$')
    return TRUE;

  return bfd_elf_is_local_label_name(abfd, name);
}

static reloc_howto_type 'elf_'___arch_name___'_howto_table'[] =
{
___reloc_howto___
};

struct ___arch_name___'_reloc_map'
{
  bfd_reloc_code_real_type bfd_reloc_val;
  unsigned int ___arch_name___'_reloc_val';
};

static const struct ___arch_name___'_reloc_map' ___arch_name___'_reloc_map'[] =
{
___reloc_map___
};

static reloc_howto_type* ___arch_name___'_reloc_type_lookup' (abfd, code)
  bfd * abfd ATTRIBUTE_UNUSED;
  bfd_reloc_code_real_type code;
{
  unsigned int i;
  for (i = ARRAY_SIZE (___arch_name___'_reloc_map'); --i;)
```

```
    if (__arch_name__'_reloc_map'[i].bfd_reloc_val == code)
        return & 'elf_'__arch_name__'_howto_table'[__arch_name__'_reloc_map'[i].
            __arch_name__'_reloc_val'];

  return NULL;
}

static void __arch_name__'_elf_info_to_howto' (abfd, cache_ptr, dst)
  bfd *abfd ATTRIBUTE_UNUSED;
  arelent *cache_ptr;
  Elf_Internal_Rela *dst;
{
  unsigned int r;
  r = ELF32_R_TYPE (dst->r_info);

  BFD_ASSERT (r < (unsigned int) 'R_'__arch_name__'_max');

  cache_ptr->howto = &'elf_'__arch_name__'_howto_table'[r];
}

bfd_reloc_status_type
'_bfd_'__arch_name__'_elf_carry_reloc' (bfd *abfd, arelent *reloc_entry, asymbol *
    symbol,
          void *data, asection *input_section, bfd *output_bfd, char **error_message)
{
  if (output_bfd == NULL) { // If it's the final link, apply the carry
    bfd_vma vallo;
    bfd_byte *location = (bfd_byte *) data + reloc_entry->address;
    vallo = bfd_get_32 (abfd, location);

    unsigned long mask1 = 1 << (32 - reloc_entry->howto->bitsize - 1);
    unsigned long mask2 = 0;
    unsigned i;
    for (i = 0; i < (32 - reloc_entry->howto->bitsize); i++)
      mask2 = mask2 | 1 << i;
```

```
    reloc_entry->addend += (vallo + mask1) & mask2;
  }

  return bfd_elf_archc_reloc (abfd, reloc_entry, symbol, data, input_section, output_bfd,
      error_message);
}


static bfd_reloc_status_type
bfd_elf_archc_reloc (bfd *abfd,
                arelent *reloc_entry,
                asymbol *symbol,
                void *data,
                asection *input_section,
                bfd *output_bfd,
                char **error_message ATTRIBUTE_UNUSED)
{
  bfd_vma relocation;
  bfd_size_type octets = reloc_entry->address * bfd_octets_per_byte (abfd);
  bfd_vma output_base = 0;
  reloc_howto_type *howto = reloc_entry->howto;
  asection *reloc_target_output_section;
// asymbol *symbol = *(reloc_entry->sym_ptr_ptr);


  /*
   * Taken from bfd_elf_generic_reloc
   * It is basicly a short circuit if the output is relocatable
   *
   * TODO: maybe print an error message if output is relocatable?
   */
  if (output_bfd != NULL
      && (symbol->flags & BSF_SECTION_SYM) == 0
      && (! reloc_entry->howto->partial_inplace
```

```
     || reloc_entry->addend == 0))
    {
      reloc_entry->address += input_section->output_offset;
      return bfd_reloc_ok;
    }

/*
 * This part is mainly copied from bfd_perform_relocation, but
 * here we use generic get/put bits
 * Field 'size' of HOWTO structure now indicates the -bit size-
 */

/* Is the address of the relocation really within the section? */
if (reloc_entry->address > bfd_get_section_limit (abfd, input_section))
  return bfd_reloc_outofrange;

/* Work out which section the relocation is targeted at and the
 * initial relocation command value. */

/* Get symbol value. (Common symbols are special.) */
if (bfd_is_com_section (symbol->section))
  relocation = 0;
else
  relocation = symbol->value;

reloc_target_output_section = symbol->section->output_section;

/* Convert input-section-relative symbol value to absolute. */
if ((output_bfd && ! howto->partial_inplace)
     || reloc_target_output_section == NULL)
  output_base = 0;
else
  output_base = reloc_target_output_section->vma;

relocation += output_base + symbol->section->output_offset;
```

```
/* Add in supplied addend. */
relocation += reloc_entry->addend;

/* Here the variable relocation holds the final address of the
 * symbol we are relocating against, plus any addend.
 */


if (howto->pc_relative) {
  relocation -= input_section->output_section->vma + input_section->
      output_offset;

  if (howto->pcrel_offset)
    relocation -= reloc_entry->address;
}

if (output_bfd != NULL) {
  /* TODO: deal with relocatable output?!? */
}
else
  reloc_entry->addend = 0;

/*
* Overflow checking is NOT being done!!!
*/



relocation >>= (bfd_vma) howto->rightshift;

/* Shift everything up to where it's going to be used. */
relocation <<= (bfd_vma) howto->bitpos;

/*
```

```
   * Now the real stuff... get, apply and put back the relocation from/into
   * object file image
   */


#define DOIT(x) \
      x = ( (x & ~howto->dst_mask) | (((x & howto->src_mask) + relocation) &
          howto->dst_mask))

  long long x;

  x = getbits(howto->size, (char *) data + octets , ___endian_val___);

  DOIT (x);

  putbits(howto->size, (char *) data + octets, x, ___endian_val___);



  return bfd_reloc_ok;
}

/*
 * 1 = big, 0 = little
 *
 * limitations:
 * . endianess affects 8-bit bytes
 * . maximum word size: 64-bit
 */
static long long getbits(unsigned int bitsize, char *location, int endian)
{
  long long data = 0;
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;
```

```
  if (bits_remain) number_bytes++;

  int index, inc;
  if (endian == 1) { /* big */
    index = 0;
    inc = 1;
  }
  else { /* little */
    index = number_bytes - 1;
    inc = -1;
  }

  while (number_bytes) {
    data = (data << 8) | (unsigned char)location[index];

    index += inc;
    number_bytes--;
  }

/*
 * if bitsize is not multiple of 8 then clear/pack the remaining bits
 */
  long long mask = 0;
  for (; bitsize; bitsize--)
    mask = (mask << 1) | 1;

  if (bits_remain) {
    if (endian == 1) {
      long long temp = data >> (8 - bits_remain);
      temp &= (mask << bits_remain);

      data = temp | (data & ~(mask << bits_remain));
    }
  }
```

```
  return data & mask;
}


static void putbits(unsigned int bitsize, char *location, long long value, int endian)
{
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;
  unsigned char bytes[8];
  unsigned int i;
  int index;
  unsigned char mask = 0;

  /*
        * Fill the bytes array so as not to depend on the host endian
        *
        * bytes[0] -> least significant byte
        */
  for (i=0; i<8; i++)
    bytes[i] = (value & (0xFF << (i*8))) >> 8*i;

  if (bits_remain) {
    for (i=0; i<bits_remain; i++)
          mask = (mask << 1) | 1;
  }

  if (endian == 0) { /* little */
    index = 0;

    while ((unsigned int)index != number_bytes) {
      location[index] = bytes[index];

      index++;
    }
```

```
        if (bits_remain)
      location[number_bytes] = (location[number_bytes] & ~mask) | (bytes[number_bytes
          ] & mask);

  }
  else { /* big */
    index = number_bytes - 1;

    i = 0;
    while (index >= 0) {
      if (bits_remain)
        location[i] = ((bytes[index+1] & mask) << (8-bits_remain)) | (bytes[index]
            >> bits_remain);
      else
        location[i] = bytes[index];

      i++;
      index--;
    }

    if (bits_remain)
      location[number_bytes] = (bytes[0] & mask) | (location[number_bytes] & ~mask);
  }
}



#define TARGET_BIG_SYM 'bfd_elf32_'___arch_name___'_vec'
#define TARGET_BIG_NAME '"elf32-'___arch_name___'"'
#define ELF_ARCH 'bfd_arch_'___arch_name___
#define ELF_MACHINE_CODE EM_NONE
#define ELF_MAXPAGESIZE 0x1
#define bfd_elf32_bfd_reloc_type_lookup ___arch_name___'_reloc_type_lookup'
#define bfd_elf32_bfd_is_local_label_name ___arch_name___'_elf_is_local_label_name'
#define elf_info_to_howto ___arch_name___'_elf_info_to_howto'
#include "elf32-target.h"
```

Listing A.19: binutils/bfd/Makefile.in.sed

```
/ALL_MACHINES = \\/ a\
        cpu-xxxxx.lo \\

/ALL_MACHINES_CFILES = \\/ a\
        cpu-xxxxx.c \\

/BFD32_BACKENDS = \\/ a\
        elf32-xxxxx.lo \\

/BFD32_BACKENDS_CFILES = \\/ a\
        elf32-xxxxx.c \\

/cpu-a29k.lo:/ i\
cpu-xxxxx.lo: cpu-xxxxx.c $(INCDIR)/filenames.h

/aout-adobe.lo:/ i\
elf32-xxxxx.lo: elf32-xxxxx.c $(INCDIR)/filenames.h elf-bfd.h \\\
  $(INCDIR)/elf/common.h $(INCDIR)/elf/internal.h $(INCDIR)/elf/external.h \\\
  $(INCDIR)/bfdlink.h $(INCDIR)/elf/xxxxx.h elf32-target.h
```

Listing A.20: binutils/bfd/targets.c.sed

```
/extern const bfd_target a29kcoff_big_vec;/ i\
extern const bfd_target bfd_elf32_xxxxx_vec;
```

Listing A.21: binutils/include/dis-asm.h.sed

```
/extern int print_insn_big_mips  (bfd_vma, disassemble_info \*);/ i\
extern int print_insn_xxxxx (bfd_vma, disassemble_info *);
```

Listing A.22: binutils/include/elf/xxxxx.h

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* ___arch_name___ ELF support for BFD.
   Copyright 2005, 2006 --- The ArchC Team.
```

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of BFD, the Binary File Descriptor library.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place − Suite 330, Boston, MA 02111−1307, USA. ∗/

```
/*
 * written by:
 * Alexandro Baldassin
 * Daniel Casarotto
 */


#ifndef '_ELF_'___arch_name___'_H_FILE_'
#define '_ELF_'___arch_name___'_H_FILE_'

#include "elf/reloc−macros.h"

/* Generic data relocation id's */
#define BFD_GENERIC_8 10008
#define BFD_GENERIC_16 10016
#define BFD_GENERIC_32 10032

#define BFD_GENERIC_REL8 10108
```

#define BFD_GENERIC_REL16 10116
#define BFD_GENERIC_REL32 10132

START_RELOC_NUMBERS (‘elf_’___arch_name___‘_reloc_type’)
___reloc_ids___
END_RELOC_NUMBERS (‘R_’___arch_name___‘_max’)

#endif

Listing A.23: binutils/include/opcode/xxxxx.h

/∗ ex: set tabstop=2 expandtab:
−∗− Mode: C; tab−width: 2; indent−tabs−mode nil −∗−
∗/
/∗ ___arch_name___‘.h.’ ___arch_name___ opcode list.
Copyright 2005, 2006 −−− The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of GDB, GAS, and the GNU binutils.

GDB, GAS, and the GNU binutils are free software; you can redistribute
them and/or modify them under the terms of the GNU General Public
License as published by the Free Software Foundation; either version
1, or (at your option) any later version.

GDB, GAS, and the GNU binutils are distributed in the hope that they
will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See
the GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this file; see the file COPYING. If not, write to the Free
Software Foundation, 59 Temple Place − Suite 330, Boston, MA 02111−1307, USA. ∗/

/∗
∗ written by:

```
 * Alexandro Baldassin
 */

#ifndef _OPC_H_FILE_
#define _OPC_H_FILE_

typedef struct {
  const char *mnemonic;
  const char *args;
  unsigned long image;
  unsigned long format_id;
  unsigned long pseudo_idx;
  unsigned long counter;
  unsigned long dmask;
} acasm_opcode;

typedef struct {
  const char *symbol;
  const char *cspec;
  unsigned long value;
} acasm_symbol;

extern long long getbits(unsigned int bitsize, char *location, int endian);
extern void putbits(unsigned int bitsize, char *location, long long value, int
    endian);

extern const int num_opcodes;
extern acasm_opcode opcodes[];
extern const acasm_symbol udsymbols[];
extern const int num_symbols;
extern const char *pseudo_instrs[];
extern const int num_pseudo_instrs;

#endif
```

Listing A.24: binutils/opcodes/configure.sed

```
/bfd_a29k_arch)/ i\
        bfd_xxxxx_arch)  ta="$ta xxxxx-dis.lo xxxxx-opc.lo" ;;
```

Listing A.25: binutils/opcodes/disassemble.c.sed

```
/#define ARCH_a29k/ i\
#define ARCH_xxxxx

/#ifdef ARCH_a29k/ i\
#ifdef ARCH_xxxxx \
    case bfd_arch_xxxxx: \
    disassemble = print_insn_xxxxx; \
    break; \
#endif
```

Listing A.26: binutils/opcodes/Makefile.in.sed

```
/CFILES = \\/ a\
        xxxxx-opc.c \\\
        xxxxx-dis.c \\

/ALL_MACHINES = \\/ a\
        xxxxx-opc.lo \\\
   xxxxx-dis.lo \\

/a29k-dis.lo:/ i\
xxxxx-opc.lo: xxxxx-opc.c sysdep.h config.h $(INCDIR)/ansidecl.h \\\
  $(INCDIR)/opcode/xxxxx.h \
xxxxx-dis.lo: xxxxx-dis.c sysdep.h config.h $(INCDIR)/ansidecl.h \\\
  $(INCDIR)/dis-asm.h $(BFD_H) $(INCDIR)/symcat.h $(INCDIR)/opcode/xxxxx.h
```

Listing A.27: binutils/opcodes/xxxxx-dis.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* Print ___arch_name___ instructions for GDB, the GNU debugger, or for objdump.
   Copyright 2005, 2006 --- The ArchC Team
```

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of GDB, GAS, and the GNU binutils.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
*Foundation, Inc., 59 Temple Place − Suite 330, Boston, MA 02111−1307, USA. */*

*/**
** written by:*
** Alexandre Keunecke I. de Mendonca*
** Felipe Guimaraes Carvalho*
** Max Ruben de Oliveira Schultz*
**/*

#include <setjmp.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdio.h>
#include "sysdep.h"
#include "dis−asm.h"
#include "opintl.h"
#include `'"opcode/'__arch_name__`.h"'

```
/*
    ----------------------------------------------------------------
    */

#define 'MAX_FORMAT_SIZE '___max_format_size___
#define 'VARIABLE_FORMAT_SIZE '___variable_format_size___
#define BI_NONE 0
#define BI_MD_REL_BIT (1 << 3)
#define BI_MD_AL_BIT (1 << 6)

/*
    ----------------------------------------------------------------
    */

// Disassembler generation definitions...

/*
  Linked list with all the operands of the isntruction.
  field_id -> position of field in the instruction, later used to know about the size of
      field, (examples: '1' or '1+2')
  type -> type of field e.g. reg, addr, immm, exp
*/
typedef struct _ac_symbol {
    char *field_id;
    char *type;
    struct _ac_symbol *next;
} ac_symbol;


struct private
{
  /* Points to first byte not fetched. */
  bfd_byte * max_fetched;
  bfd_byte the_buffer[MAX_FORMAT_SIZE];
  bfd_vma insn_start;
```

```
  jmp_buf bailout;
};

/*
    ------------------------------------------------------------
    */

/*
    bufInitial variable – set only by parse()
    Don't ever change these variables elsewhere!
*/
char bufInitial[50];

char bufFinal[50];

/*
    Addend variables – set only by get_addend()
    Don't ever change these variables elsewhere!
*/
static int valr = 0; // value used to return R addends
static int vala = 0; // value used to return A addends


/* assuming max input = 64 */
static int log_table[] = { 0 /*invalid*/, 0, 1, 1,
                           2 /* log 4 */, 2, 2, 2,
                           3 /* log 8 */, 3, 3, 3,
                           3, 3, 3, 3,
                           4 /* log 16 */, 4, 4, 4 };


/*
    ------------------------------------------------------------
    */
```

```
static int disassemble (bfd_vma memaddr, struct disassemble_info *info, unsigned
    long insn, int insn_size);

int 'print_insn_'__arch_name__' (bfd_vma memaddr, struct disassemble_info * info);'

static unsigned long get_field_size(unsigned long insn_fmt, int field_id);

static unsigned long get_insn_size(unsigned long insn_fmt);

ac_symbol* parse(char *args);

static unsigned int get_builtin_marker(char *s);

static void get_addend(char addend_type, char **st);

void replace(char *str, char *old, char *new);

/*
    ----------------------------------------------------------------------
    */


#define FETCH_DATA(info, addr) \
  ((addr) <= ((struct private *)(info->private_data))->max_fetched \
   ? 1 : fetch_data ((info), (addr)))

static int
fetch_data (struct disassemble_info *info, bfd_byte *addr)
{
  int status;
  struct private *priv = (struct private *) info->private_data;
  bfd_vma start = priv->insn_start + (priv->max_fetched - priv->the_buffer);
```

```
  status = (*info->read_memory_func) (start,
                                      priv->max_fetched,
                                      addr - priv->max_fetched,
                                      info);
  /* verify (status != 5) because for architecture with variable format size are read 8,
      16, 24 and 32 bits of memory and in the last instruction error can occur */
  if ((status != 0) && (status != 5))
    {
      (*info->memory_error_func) (status, start, info);
      longjmp (priv->bailout, 1);
    }
  else
    priv->max_fetched = addr;

  return 1;
}

static unsigned long get_field_size(unsigned long insn_fmt, int field_id)
{
___fieldsize_function___
}

static unsigned long get_insn_size(unsigned long insn_fmt)
{
___insnsize_function___
}



/*
  Function that makes the disassembler of an instruction of the object file, making the
      decoding using the field dmask of the file xxxx-opc.c

1- Using the insn apllies the mask (dmaks fiels of opcodes)
2- Search in instruction table the corresponding instruction of (insn & dmask), and
    prints the intruction mnemonic.
```

```
3– For each operand, is generated a mask in the position of the field in insn and size (e
    .g. bit 16 to the 24), after verified if is a register, or immediate, address, etc...
4– if is a register, is called the method replace to put the name of the register in the
    final string that it will be printed
5– is an immediate one!, then it verifies the modifiers, and it applies them, after calls
    the method replace to put the value in the final string(bufFinal).
6– prints string(bufFinal) in the screen with the disassembled instruction.
7– returns the octets processed, to be able to increment the address (PC)

Params
  memaddr – address of the current instruction (PC value)
  info – structure of binutils used to print the instruction and operands
  insn – the raw instruction
*/
static int disassemble (bfd_vma memaddr, struct disassemble_info *info, unsigned
    long insn, int insn_size){
  acasm_opcode *op = (acasm_opcode *) opcodes;
  acasm_symbol *symbols = (acasm_symbol *) udsymbols;
  unsigned int FORMAT_SIZE=0;

  //Find the mnemonic of instruction in opcodes table
  while (op->mnemonic){
    unsigned long e = insn & op->dmask;

    if ((e == op->image) && (op->pseudo_idx == 0) && ((unsigned)insn_size
        == get_insn_size(op->format_id))) {

      FORMAT_SIZE = get_insn_size(op->format_id);

      //Print instruction mnemonic
      (*info->fprintf_func) (info->stream,"%s\t",op->mnemonic);

      //Now Look for Registers and address
      char *args = malloc(sizeof(char)*50);
      strcpy(args, op->args);
```

```
ac_symbol* acs = parse(args);

char *ptInitial = malloc(sizeof(char)*50);
char *ptFinal = malloc(sizeof(char)*50);
ptInitial = bufInitial;
ptFinal = bufFinal;
strcpy(ptFinal,ptInitial);

while (acs){
    unsigned long begining = 0;

    char *ptfield_id = malloc(sizeof(char)*50);
    ptfield_id = acs->field_id;
    unsigned long size = 0;
    unsigned int last_field_id = 0;
    while (*ptfield_id != '\0') {
        if ((*ptfield_id >= '0') && (*ptfield_id <= '9')) {
          size = size + get_field_size(op->format_id, atoi(ptfield_id));
          last_field_id = (unsigned)atoi(ptfield_id);
        }
        ptfield_id++;
    }

    unsigned int i = 0;
    for (i=0; i<=last_field_id; i++){
        begining += get_field_size(op->format_id, i);
    }

    unsigned long dmask = 0x00;
    for (i=0; i<size; i++){
        dmask = (dmask << 1) + 1;
    }

    dmask = dmask << (FORMAT_SIZE - begining);//operand dmask
    unsigned long value = dmask & insn;
```

```
value = value >> (FORMAT_SIZE − begining); //dislocates masks to see
    it which reg ex: [0,31]
int cont = 0;
int in = 0;
char symbolaux[50];



/* Search the register table, looking value (taken from the dmask) is a
    register and finds the name of the register and prints it as operating of
    instruction, if not found, will be an immediate field or another thing it
    prints the value directly,*/

while (cont < num_symbols){
   if ((strcmp(acs−>type, symbols−>cspec) == 0) && (value ==
       symbols−>value)){
      //operand find
      in = 1;
      strcpy(symbolaux, symbols−>symbol);
   }
   symbols++;
   cont++;
}
if (in){
   char *ptr2;
   ptr2 = bufFinal;
   replace(ptr2, acs−>type, symbolaux);
}else{//not operand find, it can be: exp, immm, addr
   //verifies if has modifiers and it records in vala,and valr
   unsigned int bi_type = get_builtin_marker(acs−>type);

   int align = (bi_type & BI_MD_AL_BIT) ? 1 : 0;
   int pcrel = (bi_type & BI_MD_REL_BIT) ? 1 : 0;

   if (align){ //Modifier A
```

```
                value <<= (vala ? log_table[vala] : log_table[FORMAT_SIZE / 8]);
            }
            if (pcrel){ //Modifier R
                if (value & (1<<(size-1))){
                    value |= ((0xFFFFFFFF >> (32 - size)) & (0xFFFFFFFF << (
                        size-1)));
                    value = (((0xFFFFFFFF >> (32 - size)) & ~value) + 1);
                    value = memaddr - value + valr;
                }
                else
                    value += memaddr + valr;
            }
            char *ptr3;
            ptr3 = bufFinal;
            char new[50];
            //Convert long to char
            sprintf( new, "0x%01lx", value );
            replace(ptr3,acs->type,new);
        }
        symbols = symbols - cont;//Return table symbol pointer
        acs = acs->next;
      }
      break;
    }
    op++;
  }
  (*info->fprintf_func) (info->stream,"%s ",bufFinal);
  return FORMAT_SIZE / 8;
}


/*
  Function that is called by objdump core (entry point for the disassemble) for each
      instruction of the file, reads from the binary file the raw insn and calls the
      function disassemble() that disassemble the instruction.
```

```
*/
int 'print_insn_'___arch_name___' '(bfd_vma memaddr, struct disassemble_info *info){'
    struct private priv;
    bfd_byte *buffer = priv.the_buffer;
    unsigned long insn = 0;
    unsigned long insnfound;
    int sizeinsn;

    info->private_data = & priv;
    priv.max_fetched = priv.the_buffer;
    priv.insn_start = memaddr;

    if (setjmp (priv.bailout) != 0)
        /* Error return. */
        return -1;

    acasm_opcode *op = (acasm_opcode *) opcodes;

    /* Verify if architecture has variable format size (CISC) ou no (RISC) */
    if (VARIABLE_FORMAT_SIZE == 0) {
        /*RISC - read fix length bits of the memory */
        FETCH_DATA (info, buffer + 4);
        insn = getbits(MAX_FORMAT_SIZE, buffer, ___endian_val___);
        insnfound = insn;
        sizeinsn = MAX_FORMAT_SIZE;
    }
    else {
        /* CISC - read variable length bits of the memory
            1- read 8 bits and find instruction in opcodes table
            2- if has format size > 8, read 16 bits and find instruction in opcodes table
            3- if has format size > 16, read 24 bits and find instruction in opcodes table
            4- if has format size > 24, read 32 bits and find instruction in opcodes table
            Decode last insn find.
        */
        unsigned int localsize = 8;
```

```
        unsigned int i;
        while (localsize <= MAX_FORMAT_SIZE) {
            FETCH_DATA (info, buffer + (localsize / 8));
            for (i=3; i>=(localsize/8); i--)
                buffer[i] = 0;
            insn = getbits(localsize, buffer, ___endian_val___);

            //Find the mnemonic of instruction in opcodes table
            int j = 0;
            while ((op->mnemonic) && (op->dmask != 0)) {
                unsigned long e = insn & op->dmask;

                if ((e == op->image) && (op->pseudo_idx == 0) && (localsize ==
                     get_insn_size(op->format_id))){
                    insnfound = insn;
                    sizeinsn = localsize;
                    break;
                }
                op++;
                j++;
            }
            localsize = localsize + 8;
            op = op - j;
        }
    }

    return disassemble(memaddr, info, insnfound, sizeinsn);
}


ac_symbol* parse(char *args){
    char *ptr_bufInitial = bufInitial;
    ac_symbol *lista = NULL;
    ac_symbol *temp;
    ac_symbol *t;
```

```
t = NULL;
temp = NULL;
int i = 0;

while (*args != '\0'){
    char buf[50];
    char *ptr_buf = buf;
    char field_aux[50];
    char *ptr_field_aux = field_aux;

    /*
      '%' is the start of a well-formed sentence. It goes like this:
      '%'<conversion specifier>':'<insn field ID>':<information from ld>:
      ' %imm:5:0:,%reg:3:0:,%reg:1:0:
    */
    if (*args == '%'){
        args++;
        while (*args != ':'){
            *ptr_buf = *args;
            *ptr_bufInitial = *args;
            ptr_bufInitial++;
            ptr_buf++;
            args++;
        }

        *ptr_buf = '\0';
        args++;

        char *arg_start = args;
        while ((*args >= '0') && (*args <= '9'))
            args++;

        *ptr_field_aux = *arg_start;
        ptr_field_aux++;
        args++;
```

```
while ((*args != ':') && (*args != '+'))
    args++;

if (*args == ':') {
  args++;
}
else if (*args == '+') {
  *ptr_field_aux = *args;
  ptr_field_aux++;
  args++;
  *ptr_field_aux = *args;
  ptr_field_aux++;
  args++;
  args++;
  while (*args != ':')
    args++;
  args++;
}

*ptr_field_aux = '\0';
ptr_field_aux++;

char *to2 = malloc(sizeof(char)*50);
strcpy(to2, field_aux);

char *to = malloc(sizeof(char)*50);
strcpy(to, buf);

if (i > 0){
    temp = (ac_symbol *) malloc(sizeof(ac_symbol));
    temp->field_id = to2;
    temp->type = to;
    temp->next = NULL;
    lista->next = temp;
```

```
            lista = lista->next;
        }
        else{
            lista = (ac_symbol *) malloc(sizeof(ac_symbol));
            t = lista;
            lista->field_id = to2;
            lista->type = to;
            lista->next = NULL;
        }
        i++;
    }else{ //if (*args=='%')
        if (*args == '\\'){
            args++;
            args++;
            while ((*args>64 && *args<91) || (*args>96 && *args<123))
                args++;
        }else{
            *ptr_bufInitial = *args;
            args++;
            ptr_bufInitial++;
        }
    }
}//end of while (*args!='\0')
*ptr_bufInitial = '\0';
lista = t;
return lista;
}

/*
    ------------------------------------------------------------
    */

/*
 Bases:
  'exp'
```

```
        'addr'
        'imm'

 Modifiers:
  'H'[n][c][u|s] -> n -> get the n-high bit -- u or s (unsigned or unsigned) -- c -
      add carry from lower bits
  'L'[n][u|s] -> get the n-low bits
        'R'[n][b] -> backward, n -> add n to PC
        'A'[n][u|s] -> align in a paragraph of n-bits

 st : null-terminated string with the conversion specifier
*/
static unsigned int get_builtin_marker(char *st){
    valr = 0;
    vala = 0;

    unsigned int ret_val = BI_NONE;

    if (*st == 'e' && *(st+1) == 'x' && *(st+2) == 'p'){
        st += 3;
    }
    else if (*st == 'a' && *(st+1) == 'd' && *(st+2) == 'd' && *(st+3) == 'r'){
        st += 4;
    }
    else if (*st == 'i' && *(st+1) == 'm' && *(st+2) == 'm'){
        st += 3;
    }
    else
        return BI_NONE;

    while (*st != '\0'){
        switch (*st){
            case 'A':
                ret_val |= BI_MD_AL_BIT;
                get_addend('A', &st);
```

```
                break;
            case 'R':
                ret_val |= BI_MD_REL_BIT;
                get_addend('R', &st);
                break;
        }
        st++;
    }

    return ret_val;
}

/*
    ------------------------------------------------------------
    */

/*
   Parse and get the modifier's addend
   Only called by get_builtin_marker()

   addend_type: ''A' or 'R'

   st : string pointing to the addend type. It will be set to the next char
   after the addend
*/
static void get_addend(char addend_type, char **st){
    char *sl = *st;
    sl++;

    while (*sl >= '0' && *sl <= '9')
        sl++;

    if ((sl-1) != *st){
        char savec = *sl;
        *sl = '\0';
```

```
        if (addend_type == 'A')
            vala = atoi((*st)+1);
        else // 'R'
            valr = atoi((*st)+1);

        *sl = savec;
    }

    switch (addend_type){
        case 'R': // [b]
            if (*sl == 'b'){
              sl++;
              valr = valr * (-1);
            }
            break;
        case 'A': // [u|s]
            if (*sl == 's'){
              sl++;
            }
            else if (*sl == 'u'){
              sl++;
            }
            break;
    }

    *st = sl-1;
}


/*
    Function that from parameter 'ptr_ret' search for the first occurrence of 'old' and
        substitutes for 'new'
    e.g.: ptr_ret = addi reg,reg,exp
          old = reg
```

```
        new = $30
    result:
        ptr_ret = addi $30,reg,exp
*/
void replace(char *ptr_ret, char *old, char *new){
    char a[50];
    char *str = a;

    strcpy(str, ptr_ret);

    while (*str != '\0'){
        if (strncmp(old, str, strlen(old)) == 0){
            while(*old == *str){
                old++;
                if (*str != '\0')
                    str++;
            }
            while (*new != '\0'){
                *ptr_ret = *new;
                ptr_ret++;
                new++;
            }
            while (*str != '\0'){
                *ptr_ret = *str;
                str++;
                ptr_ret++;
            }
        }
        else{
            *ptr_ret = *str;
            str++;
            ptr_ret++;
        }
    }
```

```
    *ptr_ret = '\0';
}
```

Listing A.28: binutils/opcodes/xxxxx-opc.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* ___arch_name___'-opc.c' -- ___arch_name___ opcode list.
   Copyright 2005, 2006 --- The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of GDB, GAS, and the GNU binutils.

GDB, GAS, and the GNU binutils are free software; you can redistribute
them and/or modify them under the terms of the GNU General Public
License as published by the Free Software Foundation; either version
1, or (at your option) any later version.

GDB, GAS, and the GNU binutils are distributed in the hope that they
will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See
the GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this file; see the file COPYING. If not, write to the Free
Software Foundation, 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA. */

/*
 * written by:
 * Alexandro Baldassin
 */

#include <stdio.h>
#include "sysdep.h"
#include "'opcode/'___arch_name___'.h"'
```

```
acasm_opcode opcodes[] = {
___opcode_table___
};

const acasm_symbol udsymbols[] = {
___symbol_table___
};

const char *pseudo_instrs[] = {
___pseudo_table___
};

/*
 * 1 = big, 0 = little
 *
 * limitations:
 * . endianess affects 8-bit bytes
 * . maximum word size: 64-bit
 */
long long getbits(unsigned int bitsize, char *location, int endian)
{
  long long data = 0;
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;

  if (bits_remain) number_bytes++;

  int index, inc;
  if (endian == 1) { /* big */
    index = 0;
    inc = 1;
  }
  else { /* little */
    index = number_bytes - 1;
```

```
    inc = −1;
  }

  while (number_bytes) {
    data = (data << 8) | (unsigned char)location[index];

    index += inc;
    number_bytes−−;
  }

/*
 * if bitsize is not multiple of 8 then clear/pack the remaining bits
 */
  long long mask = 0;
  for (; bitsize; bitsize−−)
    mask = (mask << 1) | 1;

  if (bits_remain) {
    if (endian == 1) {
      long long temp = data >> (8 − bits_remain);
      temp &= (mask << bits_remain);

      data = temp | (data & ~(mask << bits_remain));
    }
  }

  return data & mask;
}


void putbits(unsigned int bitsize, char *location, long long value, int endian)
{
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;
  unsigned char bytes[8];
```

```
unsigned int i;
int index;
unsigned char mask = 0;

/*
        * Fill the bytes array so as not to depend on the host endian
        *
        * bytes[0] -> least significant byte
        */
for (i=0; i<8; i++)
  bytes[i] = (value & (0xFF << (i*8))) >> 8*i;

if (bits_remain) {
  for (i=0; i<bits_remain; i++)
          mask = (mask << 1) | 1;
}

if (endian == 0) { /* little */
  index = 0;

  while ((unsigned int)index != number_bytes) {
    location[index] = bytes[index];

    index++;
  }

        if (bits_remain)
   location[number_bytes] = (location[number_bytes] & ~mask) | (bytes[number_bytes
        ] & mask);

}
else { /* big */
  index = number_bytes - 1;

  i = 0;
```

```
    while (index >= 0) {
      if (bits_remain)
        location[i] = ((bytes[index+1] & mask) << (8−bits_remain)) | (bytes[index]
            >> bits_remain);
      else
        location[i] = bytes[index];

      i++;
      index−−;
    }

    if (bits_remain)
      location[number_bytes] = (bytes[0] & mask) | (location[number_bytes] & ~mask);
  }
}

const int num_opcodes = ((sizeof opcodes) / (sizeof(opcodes[0])));
const int num_symbols = ((sizeof udsymbols) / (sizeof(udsymbols[0])));
const int num_pseudo_instrs = ((sizeof pseudo_instrs) / (sizeof(pseudo_instrs[0])));
```

Listing A.29: gdb/config.sub.sed

```
/1750a | 580 \\/ a\
        | xxxxx \\

/580−\* \\/ a\
        | xxxxx−* \\

/\*−acorn)/ i\
        xxxxx−*) \
                os=−elf \
                ;;
```

Listing A.30: gdb/bfd/archures.c.sed

```
/extern const bfd_arch_info_type bfd_alpha_arch;/ i\
extern const bfd_arch_info_type bfd_xxxxx_arch;
```

Listing A.31: gdb/bfd/bfd-in2.h.sed

```
/bfd_arch_last/ i\
  bfd_arch_xxxxx,
```

Listing A.32: gdb/bfd/config.bfd.sed

```
/alpha\*)/ i\
xxxxx)  targ_archs=bfd_xxxxx_arch ;;

/am33_2.0-\*-linux\*)/ i\
  xxxxx-*-elf) \
    targ_defvec=bfd_elf32_xxxxx_vec \
    ;;
```

Listing A.33: gdb/bfd/configure.sed

```
/a_out_adobe_vec)/ i\
    bfd_elf32_xxxxx_vec) tb="$tb elf32-xxxxx.lo elf32.lo $elf" ;;
```

Listing A.34: gdb/bfd/cpu-xxxxx.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* bfd back-end for ___arch_name___ support
   Copyright 2005, 2006 --- The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of BFD, the Binary File Descriptor library.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place − Suite 330, Boston, MA 02111−1307, USA. */

/*
 * written by:
 * Alexandro Baldassin
 */

#include "bfd.h"
#include "sysdep.h"
#include "libbfd.h"

const 'bfd_arch_info_type bfd_'___arch_name___'_arch' = {
  ___word_size___, /* bits in a word. */
  32, /* bits in an address. */
  8, /* bits in a byte. */
  'bfd_arch_'___arch_name___, /* enum bfd_architecture arch. */
  0, /* machine number */
  "'___arch_name___"', /* arch_name. */
  "'___arch_name___"', /* printable name. */
  3, /* unsigned int section alignment power. */
  TRUE, /* the one and only. */
  bfd_default_compatible,
  bfd_default_scan,
  NULL
};
```

Listing A.35: gdb/bfd/elf32-xxxxx.c

```
/* ex: set tabstop=2 expandtab:
   −*− Mode: C; tab−width: 2; indent−tabs−mode nil −*−
*/
/* ___arch_name___'−specific' support for ELF
   Copyright 2005, 2006 −−− The ArchC Team
```

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of BFD, the Binary File Descriptor library.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
*Foundation, Inc., 59 Temple Place − Suite 330, Boston, MA 02111−1307, USA. */*

```
/*
 * written by:
 * Alexandro Baldassin
 * Daniel Casarotto
 */

#include "bfd.h"
#include "sysdep.h"
#include "libbfd.h"
#include "elf−bfd.h"
#include '"elf/'__arch_name__'.h"'
#include "libiberty.h"

/*
 BFD prior to version 2.16 does not include bfd_get_section_limit
*/
#ifndef bfd_get_section_limit
#define bfd_get_section_limit(bfd, sec) \
```

```
  (((sec)->_cooked_size) \
   / bfd_octets_per_byte (bfd))
#endif

static reloc_howto_type* ___arch_name___'_reloc_type_lookup'(bfd*,
    bfd_reloc_code_real_type);
static void ____arch_name___'_elf_info_to_howto' (bfd *, arelent *, Elf_Internal_Rela *);
static long long getbits(unsigned int bitsize, char *location, int endian);
static void putbits(unsigned int bitsize, char *location, long long value, int endian)
    ;


/* ArchC generic relocation routine prototype*/
static bfd_reloc_status_type
bfd_elf_archc_reloc (bfd *abfd,
                     arelent *reloc_entry,
                     asymbol *symbol,
                     void *data,
                     asection *input_section,
                     bfd *output_bfd,
                     char **error_message ATTRIBUTE_UNUSED);


/* carry relocation function prototype */
bfd_reloc_status_type
'_bfd_'___arch_name___'_elf_carry_reloc' (bfd *abfd, arelent *reloc_entry, asymbol *symbol
    ,
            void *data, asection *input_section, bfd *output_bfd, char **error_message)
                ;

/* Local label (those who starts with $ are recognized as well) */
static bfd_boolean
___arch_name___'_elf_is_local_label_name'(bfd*abfd,const_char*name)
{
  if(name[0]=='$')
```

```
    return TRUE;

  return bfd_elf_is_local_label_name(abfd, name);
}

static reloc_howto_type 'elf_'___arch_name___'_howto_table'[] =
{
___reloc_howto___
};

struct ___arch_name___'_reloc_map'
{
  bfd_reloc_code_real_type bfd_reloc_val;
  unsigned int ___arch_name___'_reloc_val';
};

static const struct ___arch_name___'_reloc_map' ___arch_name___'_reloc_map'[] =
{
___reloc_map___
};

static reloc_howto_type* ___arch_name___'_reloc_type_lookup' (abfd, code)
  bfd * abfd ATTRIBUTE_UNUSED;
  bfd_reloc_code_real_type code;
{
  unsigned int i;
  for (i = ARRAY_SIZE (___arch_name___'_reloc_map'); --i;)
    if (___arch_name___'_reloc_map'[i].bfd_reloc_val == code)
      return & 'elf_'___arch_name___'_howto_table'[___arch_name___'_reloc_map'[i].
          ___arch_name___'_reloc_val'];

  return NULL;
}

static void ___arch_name___'_elf_info_to_howto' (abfd, cache_ptr, dst)
```

```
  bfd *abfd ATTRIBUTE_UNUSED;
  arelent *cache_ptr;
  Elf_Internal_Rela *dst;
{
  unsigned int r;
  r = ELF32_R_TYPE (dst->r_info);

  BFD_ASSERT (r < (unsigned int) 'R_'___arch_name___'_max');

  cache_ptr->howto = &'elf_'___arch_name___'_howto_table'[r];
}

bfd_reloc_status_type
'_bfd_'___arch_name___'_elf_carry_reloc' (bfd *abfd, arelent *reloc_entry, asymbol *
    symbol,
          void *data, asection *input_section, bfd *output_bfd, char **error_message)
{
  if (output_bfd == NULL) { // If it's the final link, apply the carry
     bfd_vma vallo;
     bfd_byte *location = (bfd_byte *) data + reloc_entry->address;
     vallo = bfd_get_32 (abfd, location);

     unsigned long mask1 = 1 << (32 - reloc_entry->howto->bitsize - 1);
     unsigned long mask2 = 0;
     unsigned i;
     for (i = 0; i < (32 - reloc_entry->howto->bitsize); i++)
       mask2 = mask2 | 1 << i;

     reloc_entry->addend += (vallo + mask1) & mask2;
   }

   return bfd_elf_archc_reloc (abfd, reloc_entry, symbol, data, input_section, output_bfd,
        error_message);
}
```

```
static bfd_reloc_status_type
bfd_elf_archc_reloc (bfd *abfd,
                     arelent *reloc_entry,
                     asymbol *symbol,
                     void *data,
                     asection *input_section,
                     bfd *output_bfd,
                     char **error_message ATTRIBUTE_UNUSED)
{
  bfd_vma relocation;
  bfd_size_type octets = reloc_entry->address * bfd_octets_per_byte (abfd);
  bfd_vma output_base = 0;
  reloc_howto_type *howto = reloc_entry->howto;
  asection *reloc_target_output_section;
// asymbol *symbol = *(reloc_entry->sym_ptr_ptr);


  /*
   * Taken from bfd_elf_generic_reloc
   * It is basicly a short circuit if the output is relocatable
   *
   * TODO: maybe print an error message if output is relocatable?
   */
  if (output_bfd != NULL
      && (symbol->flags & BSF_SECTION_SYM) == 0
      && (! reloc_entry->howto->partial_inplace
      || reloc_entry->addend == 0))
     {
       reloc_entry->address += input_section->output_offset;
       return bfd_reloc_ok;
     }

  /*
   * This part is mainly copied from bfd_perform_relocation, but
```

```
 * here we use generic get/put bits
 * Field 'size' of HOWTO structure now indicates the -bit size-
 */

/* Is the address of the relocation really within the section? */
if (reloc_entry->address > bfd_get_section_limit (abfd, input_section))
  return bfd_reloc_outofrange;

/* Work out which section the relocation is targeted at and the
 * initial relocation command value. */

/* Get symbol value. (Common symbols are special.) */
if (bfd_is_com_section (symbol->section))
  relocation = 0;
else
  relocation = symbol->value;

reloc_target_output_section = symbol->section->output_section;

/* Convert input-section-relative symbol value to absolute. */
if ((output_bfd && ! howto->partial_inplace)
    || reloc_target_output_section == NULL)
  output_base = 0;
else
  output_base = reloc_target_output_section->vma;

relocation += output_base + symbol->section->output_offset;

/* Add in supplied addend. */
relocation += reloc_entry->addend;

/* Here the variable relocation holds the final address of the
 * symbol we are relocating against, plus any addend.
 */
```

```
  if (howto->pc_relative) {
    relocation -= input_section->output_section->vma + input_section->
        output_offset;

    if (howto->pcrel_offset)
      relocation -= reloc_entry->address;
  }

  if (output_bfd != NULL) {
    /* TODO: deal with relocatable output?!? */
  }
  else
    reloc_entry->addend = 0;

 /*
  * Overflow checking is NOT being done!!!
  */



  relocation >>= (bfd_vma) howto->rightshift;

  /* Shift everything up to where it's going to be used. */
  relocation <<= (bfd_vma) howto->bitpos;

  /*
   * Now the real stuff... get, apply and put back the relocation from/into
   * object file image
   */


#define DOIT(x) \
      x = ( (x & ~howto->dst_mask) | (((x & howto->src_mask) + relocation) &
          howto->dst_mask))
```

```
  long long x;

  x = getbits(howto->size, (char *) data + octets , ___endian_val___);

  DOIT (x);

  putbits(howto->size, (char *) data + octets, x, ___endian_val___);



  return bfd_reloc_ok;
}

/*
 * 1 = big, 0 = little
 *
 * limitations:
 * . endianess affects 8-bit bytes
 * . maximum word size: 64-bit
 */
static long long getbits(unsigned int bitsize, char *location, int endian)
{
  long long data = 0;
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;

  if (bits_remain) number_bytes++;

  int index, inc;
  if (endian == 1) { /* big */
    index = 0;
    inc = 1;
  }
  else { /* little */
```

```
    index = number_bytes − 1;
    inc = −1;
  }

  while (number_bytes) {
    data = (data << 8) | (unsigned char)location[index];

    index += inc;
    number_bytes−−;
  }

/*
 * if bitsize is not multiple of 8 then clear/pack the remaining bits
 */
  long long mask = 0;
  for (; bitsize; bitsize−−)
    mask = (mask << 1) | 1;

  if (bits_remain) {
    if (endian == 1) {
      long long temp = data >> (8 − bits_remain);
      temp &= (mask << bits_remain);

      data = temp | (data & ~(mask << bits_remain));
    }
  }

  return data & mask;
}


static void putbits(unsigned int bitsize, char *location, long long value, int endian)
{
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;
```

```
unsigned char bytes[8];
unsigned int i;
int index;
unsigned char mask = 0;

/*
      * Fill the bytes array so as not to depend on the host endian
      *
      * bytes[0] -> least significant byte
      */
for (i=0; i<8; i++)
  bytes[i] = (value & (0xFF << (i*8))) >> 8*i;

if (bits_remain) {
  for (i=0; i<bits_remain; i++)
        mask = (mask << 1) | 1;
}

if (endian == 0) { /* little */
  index = 0;

  while ((unsigned int)index != number_bytes) {
    location[index] = bytes[index];

    index++;
  }

      if (bits_remain)
   location[number_bytes] = (location[number_bytes] & ~mask) | (bytes[number_bytes
      ] & mask);

}
else { /* big */
  index = number_bytes - 1;
```

```
    i = 0;
    while (index >= 0) {
      if (bits_remain)
        location[i] = ((bytes[index+1] & mask) << (8−bits_remain)) | (bytes[index]
            >> bits_remain);
      else
        location[i] = bytes[index];

      i++;
      index−−;
    }

    if (bits_remain)
      location[number_bytes] = (bytes[0] & mask) | (location[number_bytes] & ˜mask);
  }
}


#define TARGET_BIG_SYM ‘bfd_elf32_’___arch_name___‘_vec’
#define TARGET_BIG_NAME ‘”elf32−’___arch_name___‘”’
#define ELF_ARCH ‘bfd_arch_’___arch_name___
#define ELF_MACHINE_CODE EM_NONE
#define ELF_MAXPAGESIZE 0x1
#define bfd_elf32_bfd_reloc_type_lookup ___arch_name___‘_reloc_type_lookup’
#define bfd_elf32_bfd_is_local_label_name ___arch_name___‘_elf_is_local_label_name’
#define elf_info_to_howto ___arch_name___‘_elf_info_to_howto’
#include ”elf32−target.h”
```

Listing A.36: gdb/bfd/Makefile.in.sed

```
/ALL_MACHINES = \\/ a\
        cpu−xxxxx.lo \\

/ALL_MACHINES_CFILES = \\/ a\
        cpu−xxxxx.c \\

/BFD32_BACKENDS = \\/ a\
```

```
        elf32−xxxxx.lo \\

/BFD32_BACKENDS_CFILES = \\/ a\
        elf32−xxxxx.c \\

/cpu−alpha.lo:/ i\
cpu−xxxxx.lo: cpu−xxxxx.c $(INCDIR)/filenames.h $(INCDIR)/hashtab.h

/aout−adobe.lo:/ i\
elf32−xxxxx.lo: elf32−xxxxx.c $(INCDIR)/filenames.h $(INCDIR)/hashtab.h \\\
  $(INCDIR)/bfdlink.h genlink.h elf−bfd.h $(INCDIR)/elf/common.h \\\
  $(INCDIR)/elf/internal.h $(INCDIR)/elf/external.h \\\
  $(INCDIR)/elf/reloc−macros.h $(INCDIR)/elf/xxxxx.h elf32−target.h
```

Listing A.37: gdb/bfd/targets.c.sed

```
/extern const bfd_target a_out_adobe_vec;/ i\
extern const bfd_target bfd_elf32_xxxxx_vec;
```

Listing A.38: gdb/gdb/configure.host.sed

```
/case "${host_cpu}" in/ a\
xxxxx*)   gdb_host_cpu=xxxxx ;;
```

Listing A.39: gdb/gdb/configure.tgt.sed

```
/case "${target_cpu}" in/ a\
xxxxx*)   gdb_target_cpu=xxxxx ;;

/gdb_target=alpha−linux ;;/ a\
xxxxx−*−*)  gdb_target=xxxxx ;;
```

Listing A.40: gdb/gdb/Makefile.in.sed

```
/ALLDEPFILES = \\/ a\
        xxxxx−tdep.c \\

/abug−rom.o:/ i\
xxxxx−tdep.o: xxxxx−tdep.c $(defs_h) $(frame_h) $(frame_unwind_h) \\\
        $(frame_base_h) $(trad_frame_h) $(gdbcmd_h) $(gdbcore_h) \\\
```

```
        $(inferior_h) $(symfile_h) $(arch_utils_h) $(regcache_h) \\\
        $(gdb_string_h) $(dis_asm_h)
```

Listing A.41: gdb/gdb/xxxxx-tdep.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* Initialize ___arch_name___ target dependent functions for GDB, the GNU debugger.
   Copyright 2005, 2006 --- The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of GDB.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA. */

/*
 * written by:
 * Alexandre Keunecke I. de Mendonca
 * Felipe Guimaraes Carvalho
 * Max Ruben de Oliveira Schultz
 */

#include "defs.h"
#include "frame.h"
```

```
#include "frame-base.h"
#include "trad-frame.h"
#include "frame-unwind.h"
#include "dwarf2-frame.h"
#include "gdbtypes.h"
#include "inferior.h"
#include "gdb_string.h"
#include "gdb_assert.h"
#include "gdbcore.h"
#include "arch-utils.h"
#include "regcache.h"
#include "dis-asm.h"
#include "osabi.h"
#include `"opcode/'__arch_name__`.h"'

//TODO - GERAR A PARTIR DO MODELO
#define NUM_REGISTERS 90
#define NUM_REGISTER_SP 29
#define NUM_REGISTER_FP 30
#define NUM_REGISTER_PC 37
#define NUM_REGISTER_RET1 31
#define NUM_REGISTER_RET2 -1

#define OFFSET 0



#define X_OP(i) (((i) >> 30) & 0x3)
#define X_OP3(i) (((i) >> 19) & 0x3f)

enum
{
  `'__arch_name__`_INSN32_SIZE = 4,'
};
```

```
/*
    ----------------------------------------------------------------
    */

struct ``__arch_name__`_frame_cache'
{
  CORE_ADDR base;
  struct trad_frame_saved_reg *saved_regs;
  int frameless_p;
};

/*
    ----------------------------------------------------------------
    */

static CORE_ADDR
``__arch_name__`_skip_prologue (CORE_ADDR pc);'

static CORE_ADDR
``__arch_name__`_scan_prologue (CORE_ADDR start_pc, CORE_ADDR limit_pc,'
  struct frame_info *next_frame,
  struct ``__arch_name__`_frame_cache *this_cache);'

static ULONGEST
``__arch_name__`_fetch_instruction (CORE_ADDR addr);'

static const gdb_byte *
``__arch_name__`_breakpoint_from_pc (CORE_ADDR *pcptr, int *lenptr);'

static CORE_ADDR
read_next_frame_reg (struct frame_info *, int);

static void
set_reg_offset (struct ``__arch_name__`_frame_cache *this_cache, int regnum,'
    CORE_ADDR offset);
```

```
static struct type *
''___arch_name___'_register_type (struct gdbarch *gdbarch, int reg_num);'

static CORE_ADDR
''___arch_name___'_unwind_pc (struct gdbarch *gdbarch, struct frame_info *next_frame);
    '


static const struct frame_unwind *
''___arch_name___'_frame_sniffer (struct frame_info *next_frame);'

static void
''___arch_name___'_frame_this_id (struct frame_info *next_frame, void **this_cache,'
  struct frame_id *this_id);

static void
''___arch_name___'_frame_prev_register (struct frame_info *next_frame, void **this_cache
    ,'
  int regnum, int *optimizedp,
  enum lval_type *lvalp, CORE_ADDR *addrp,
  int *realnump, void *valuep);

static struct ''___arch_name___'_frame_cache *'
''___arch_name___'_frame_cache (struct frame_info *next_frame, void **this_cache);'

static const char *
''___arch_name___'_register_name (int reg_num);'

static struct frame_id
''___arch_name___'_unwind_dummy_id (struct gdbarch *gdbarch, struct frame_info *
    next_frame);'

static const struct frame_base *
''___arch_name___'_frame_base_sniffer (struct frame_info *next_frame);'
```

```
static CORE_ADDR
''__arch_name__`_frame_base_address (struct frame_info *next_frame, void **this_cache
    );'

/*
    ----------------------------------------------------------------
    */

/* To skip prologues, I use this predicate. Returns either PC itself
   if the code at PC does not look like a function prologue; otherwise
   returns an address that (if we're lucky) follows the prologue. If
   LENIENT, then we must skip everything which is involved in setting
   up the frame (it's OK to skip more, just so long as we don't skip
   anything which might clobber the registers which are being saved.
   We must skip more in the case where part of the prologue is in the
   delay slot of a non-prologue instruction). */
static CORE_ADDR
''__arch_name__`_skip_prologue (CORE_ADDR pc)'
{
  CORE_ADDR limit_pc;
  CORE_ADDR func_addr;

  /* See if we can determine the end of the prologue via the symbol table.
     If so, then return either PC, or the PC after the prologue, whichever is greater. */
  if (find_pc_partial_function (pc, NULL, &func_addr, NULL)) {
    CORE_ADDR post_prologue_pc = skip_prologue_using_sal (func_addr);
    if (post_prologue_pc != 0) {
      return max (pc, post_prologue_pc);
    }
  }
}

/* Fetch and return instruction from the specified location. */
static ULONGEST
''__arch_name__`_fetch_instruction (CORE_ADDR addr)'
```

```
{
  'gdb_byte buf['__arch_name__'_INSN32_SIZE];'
  int instlen;
  int status;

  instlen = ''__arch_name__'_INSN32_SIZE;'
  status = deprecated_read_memory_nobpt (addr, buf, instlen);
  if (status)
    memory_error (status, addr);
  return extract_unsigned_integer (buf, instlen);
}

/* This function implements the BREAKPOINT_FROM_PC macro.
   Use the program counter to determine the contents and size of a breakpoint
       instruction.
   It returns a pointer to a string of bytes that encode a breakpoint instruction, stores
   the length of the string to *lenptr, and adjusts pc (if necessary) to point to the actual
   memory location where the breakpoint should be inserted. */
static const gdb_byte *
''__arch_name__'_breakpoint_from_pc_(CORE_ADDR_*pcptr,_int_*lenptr)'
{
  //TODO - VER O QUE FAZER POIS PARECE QUE NAO PRECISA
      ??????????????????????????
  //static gdb_byte big_breakpoint[] = { 0, 0x5, 0, 0xd };
  //*lenptr = sizeof (big_breakpoint);
  //return big_breakpoint;
}

static CORE_ADDR
read_next_frame_reg (struct frame_info *fi, int regno)
{
  if (fi == NULL) {
    LONGEST val;
    regcache_cooked_read_signed (current_regcache, regno, &val);
    return val;
```

```
  }
  else
    return frame_unwind_register_signed (fi, regno);
}

/* Set a register's saved stack address in temp_saved_regs. If an
   address has already been set for this register, do nothing; this
   way we will only recognize the first save of a given register in a
   function prologue.

   For simplicity, save the address in both [0 .. NUM_REGS) and
   [NUM_REGS .. 2*NUM_REGS). Strictly speaking, only the second range
   is used as it is only second range (the ABI instead of ISA
   registers) that comes into play when finding saved registers in a
   frame. */
static void
set_reg_offset (struct ''__arch_name__'_frame_cache *this_cache,'int regnum,'
    CORE_ADDR offset)
{
  if (this_cache != NULL && this_cache->saved_regs[regnum].addr == -1) {
    this_cache->saved_regs[regnum + 0 * NUM_REGS].addr = offset;
    this_cache->saved_regs[regnum + 1 * NUM_REGS].addr = offset;
  }
}

/* Return the GDB type object for the "standard" data type of data in register "regnum
    ". */
static struct type *
''__arch_name__'_register_type(struct gdbarch *gdbarch,'int regnum)'
{
  //TODO - GERAR A PARTIR DO MODELO, TYPES ESTÃO EM gdbtypes.c
  return builtin_type_int32;
}

/* Given THIS_FRAME, find the previous frame's resume PC (which will be used
```

```
   to construct the previous frame's ID, after looking up the containing function). */
static CORE_ADDR
''___arch_name___'_unwind_pc_(struct_gdbarch_*gdbarch,_struct_frame_info_*next_frame)'
{
  return frame_unwind_register_signed (next_frame, NUM_REGISTER_PC);
}

/* Given a GDB frame, determine the address of the calling function's frame.
   This will be used to create a new GDB frame struct. */
static void
''___arch_name___'_frame_this_id_(struct_frame_info_*next_frame,_void_**this_cache,'
  struct frame_id *this_id)
{
  'struct '___arch_name___'_frame_cache_*info_=_' ''___arch_name___'_frame_cache_(
      next_frame,_this_cache);'
  (*this_id) = frame_id_build (info->base, frame_func_unwind (next_frame));
}

'static const struct frame_unwind '___arch_name___'_frame_unwind_=_{'
  NORMAL_FRAME,
  ''___arch_name___'_frame_this_id,'
  ''___arch_name___'_frame_prev_register'
};

static const struct frame_unwind *
''___arch_name___'_frame_sniffer_(struct_frame_info_*next_frame)'
{
  CORE_ADDR pc = frame_pc_unwind (next_frame);
  return '&'___arch_name___'_frame_unwind;'
}

/* Return the name of the register corresponding to "regnum". */
static const char *
''___arch_name___'_register_name_(int_regnum)'
{
```

```
  acasm_symbol *symbols = (acasm_symbol *) udsymbols;

  int cont = 0;
  int find_reg_name = 0;
  while (cont < num_symbols) {
    if (regnum == symbols->value) { //reg_name find
      find_reg_name = 1;
      break;
    }
    symbols++;
    cont++;
  }

  if (find_reg_name) {
    return symbols->symbol;
  }
  else {
    static char ret[5];
    sprintf(ret, "$%i\0", regnum);
    return ret;
  }
}

static struct frame_id
``__arch_name__`_unwind_dummy_id_(struct_gdbarch_*gdbarch,_struct_frame_info_*
    next_frame)'
{
  return frame_id_build (frame_unwind_register_signed (next_frame,
      NUM_REGISTER_SP),
                         frame_pc_unwind (next_frame));
}

static CORE_ADDR
``__arch_name__`_frame_base_address_(struct_frame_info_*next_frame,_void_**this_cache
    )'
```

```
{
  `struct '___arch_name___`_frame_cache␣*info␣=␣' ``___arch_name___`_frame_cache␣(
      next_frame,␣this_cache);'
  return info->base;
}

static const struct frame_base ``___arch_name___`_frame_base␣='
{
  `&'___arch_name___`_frame_unwind,'
  ``___arch_name___`_frame_base_address,'
  ``___arch_name___`_frame_base_address,'
  ``___arch_name___`_frame_base_address'
};

static const struct frame_base *
``___arch_name___`_frame_base_sniffer␣(struct␣frame_info␣*next_frame)'
{
  if `('___arch_name___`_frame_sniffer␣(next_frame)␣!=␣NULL)'
    return `&'___arch_name___`_frame_base;'
  else
    return NULL;
}

/* Put here the code to store, into fi->saved_regs, the addresses of
   the saved registers of frame described by FRAME_INFO. This
   includes special registers such as pc and fp saved in special ways
   in the stack frame. sp is even more special: the address we return
   for it IS the sp for the next frame. */
static struct ``___arch_name___`_frame_cache␣*'
``___arch_name___`_frame_cache␣(struct␣frame_info␣*next_frame,␣void␣**this_cache)'
{
  struct ``___arch_name___`_frame_cache␣*cache;'

  if ((*this_cache) != NULL)
    return (*this_cache);
```

```
  cache = FRAME_OBSTACK_ZALLOC (struct ''__arch_name__'_frame_cache);'
  cache->frameless_p=1;
  (*this_cache) = cache;
  cache->saved_regs = trad_frame_alloc_saved_regs (next_frame);

  /* Analyze the function prologue. */
  const CORE_ADDR pc = frame_pc_unwind (next_frame);
  CORE_ADDR start_addr;

  find_pc_partial_function (pc, NULL, &start_addr, NULL);
  /* We can't analyze the prologue if we couldn't find the begining of the function. */
  if (start_addr == 0)
    return cache;

  ''__arch_name__'_scan_prologue (start_addr, pc, next_frame, *this_cache);'

  if (cache->frameless_p)
    cache->base = frame_unwind_register_unsigned (next_frame, NUM_REGISTER_SP
        );
  else
    cache->base = frame_unwind_register_unsigned (next_frame, NUM_REGISTER_FP
        );

  return (*this_cache);
}


static void
''__arch_name__'_frame_prev_register (struct frame_info *next_frame, void **this_cache
    ,'
  int regnum, int *optimizedp,
  enum lval_type *lvalp, CORE_ADDR *addrp,
  int *realnump, void *valuep)
{
```

```
  `struct '__arch_name__`_frame_cache *info=' ``__arch_name__`_frame_cache (
      next_frame, this_cache);'

  if (regnum == NUM_REGISTER_PC) {
    if (valuep) {
      CORE_ADDR pc = 0;
      regnum = info->frameless_p ? NUM_REGISTER_RET1 :
          NUM_REGISTER_RET2;
      pc += frame_unwind_register_unsigned (next_frame, regnum) + OFFSET + 4;
      store_unsigned_integer (valuep, 4, pc);
    }
   if (NUM_REGISTER_RET2 != -1)
     return; //only sparc
  }

  trad_frame_get_prev_register (next_frame, info->saved_regs, regnum, optimizedp,
      lvalp, addrp, realnump, valuep);
}

/* Analyze the function prologue from START_PC to LIMIT_PC.
   Builds the associated FRAME_CACHE if not null.
   Return the address of the first instruction past the prologue. */
static CORE_ADDR
``__arch_name__`_scan_prologue (CORE_ADDR start_pc, CORE_ADDR limit_pc,'
  struct frame_info *next_frame,
  struct ``__arch_name__`_frame_cache *this_cache)'
{
  CORE_ADDR cur_pc;
  CORE_ADDR sp;

  /* Can be called when there's no process, and hence when there's no NEXT_FRAME.
       */
  if (next_frame != NULL)
    sp = read_next_frame_reg (next_frame, NUM_REGISTER_SP);
  else
```

```
    sp = 0;

  if (start_pc < limit_pc) {
    unsigned long inst;

    /* Fetch the instruction. */
    inst = (unsigned long) ''__arch_name__'_fetch_instruction_(start_pc);'

    /* Check for the SAVE instruction that sets up the frame, only sparc. */
    if ((NUM_REGISTER_RET2 != -1) && (X_OP (inst) == 2 && X_OP3 (inst) ==
        0x3c))
      this_cache->frameless_p = 0;

    set_reg_offset (this_cache, NUM_REGISTER_RET1, limit_pc+OFFSET+4);
  }

  if (this_cache != NULL) {
    this_cache->base = (frame_unwind_register_signed (next_frame,
        NUM_REGISTER_SP));
    this_cache->saved_regs[NUM_REGISTER_PC] = this_cache->saved_regs[
        NUM_REGISTER_RET1];
  }
}

/* linking between gdb core functions and local functions */
static struct gdbarch *
''__arch_name__'_gdbarch_init_(struct_gdbarch_info_info,_struct_gdbarch_list_*arches)'
{
  struct gdbarch *gdbarch;

  if (info.bfd_arch_info->arch != 'bfd_arch_'__arch_name__')'
    return NULL;

  gdbarch = gdbarch_alloc (&info, NULL);
```

```
/* Information about registers */
set_gdbarch_num_regs (gdbarch, NUM_REGISTERS);
set_gdbarch_sp_regnum (gdbarch, NUM_REGISTER_SP);
set_gdbarch_pc_regnum (gdbarch, NUM_REGISTER_PC);
set_gdbarch_register_name (gdbarch, ''___arch_name___'_register_name);'
set_gdbarch_register_type (gdbarch, ''___arch_name___'_register_type);'
set_gdbarch_num_pseudo_regs (gdbarch, 0);

/* The stack grows downward. */
set_gdbarch_inner_than (gdbarch, core_addr_lessthan);

/* Advance PC across function entry code. */
set_gdbarch_skip_prologue (gdbarch, ''___arch_name___'_skip_prologue);'

/* Breakpoint manipulation. */
set_gdbarch_breakpoint_from_pc (gdbarch, ''___arch_name___'_breakpoint_from_pc);'

/* Disassembly. */
set_gdbarch_print_insn (gdbarch, print_insn_''___arch_name___');'

/* Add some default predicates. */
frame_unwind_append_sniffer (gdbarch, ''___arch_name___'_frame_sniffer);'
frame_base_append_sniffer (gdbarch, ''___arch_name___'_frame_base_sniffer);'

/* Frame handling. */
set_gdbarch_unwind_pc (gdbarch, ''___arch_name___'_unwind_pc);'
set_gdbarch_unwind_dummy_id (gdbarch, ''___arch_name___'_unwind_dummy_id);'

set_gdbarch_short_bit (gdbarch, 2 * TARGET_CHAR_BIT);
set_gdbarch_int_bit (gdbarch, 4 * TARGET_CHAR_BIT);
set_gdbarch_long_bit (gdbarch, 4 * TARGET_CHAR_BIT);
set_gdbarch_long_long_bit (gdbarch, 8 * TARGET_CHAR_BIT);
set_gdbarch_float_bit (gdbarch, 4 * TARGET_CHAR_BIT);
set_gdbarch_double_bit (gdbarch, 8 * TARGET_CHAR_BIT);
set_gdbarch_long_double_bit (gdbarch, 8 * TARGET_CHAR_BIT);
```

```
  set_gdbarch_ptr_bit (gdbarch, 4 * TARGET_CHAR_BIT);
  set_gdbarch_addr_bit (gdbarch, 4 * TARGET_CHAR_BIT);

  /* Hook in ABI-specific overrides, if they have been registered. */
  gdbarch_init_osabi (info, gdbarch);

  return gdbarch;
}

extern 'initialize_file_ftype _initialize_'___arch_name___'_tdep;' /* -Wmissing-prototypes
    */

/* Start of target-dependent code for the architecture */
void
'_initialize_'___arch_name___'_tdep_(void)'
{
  register_gdbarch_init ('bfd_arch_'___arch_name___',' ''___arch_name___'_gdbarch_init);'
}

/*
   ----------------------------------------------------------------------
   */
```

Listing A.42: gdb/gdb/config/xxxxx/xxxxx.mt

```
# Target: ___arch_name___
TDEPFILES= ___arch_name___-tdep.o
```

Listing A.43: gdb/include/dis-asm.h.sed

```
/extern int print_insn_big_mips  (bfd_vma, disassemble_info \*);/ i\
extern int print_insn_xxxxx (bfd_vma, disassemble_info *);
```

Listing A.44: gdb/include/elf/xxxxx.h

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* ___arch_name___ ELF support for BFD.
```

```
/*
 * written by:
 * Alexandro Baldassin
 * Daniel Casarotto
 */


#ifndef '_ELF_'___arch_name___'_H_FILE_'
#define '_ELF_'___arch_name___'_H_FILE_'

#include "elf/reloc−macros.h"

/* Generic data relocation id's */
#define BFD_GENERIC_8 10008
#define BFD_GENERIC_16 10016
#define BFD_GENERIC_32 10032
```

#define BFD_GENERIC_REL8 10108
#define BFD_GENERIC_REL16 10116
#define BFD_GENERIC_REL32 10132

START_RELOC_NUMBERS (‘elf_’___arch_name___‘_reloc_type’)
___reloc_ids___
END_RELOC_NUMBERS (‘R_’___arch_name___‘_max’)

#endif

Listing A.45: gdb/include/opcode/xxxxx.h

/ ex: set tabstop=2 expandtab:
−*− Mode: C; tab−width: 2; indent−tabs−mode nil −*−
/
/ ___arch_name___‘.h.’ ___arch_name___ opcode list.
Copyright 2005, 2006 −−− The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of GDB, GAS, and the GNU binutils.

GDB, GAS, and the GNU binutils are free software; you can redistribute
them and/or modify them under the terms of the GNU General Public
License as published by the Free Software Foundation; either version
1, or (at your option) any later version.

GDB, GAS, and the GNU binutils are distributed in the hope that they
will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See
the GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this file; see the file COPYING. If not, write to the Free
*Software Foundation, 59 Temple Place − Suite 330, Boston, MA 02111−1307, USA. */

```
/*
 * written by:
 * Alexandro Baldassin
 */

#ifndef _OPC_H_FILE_
#define _OPC_H_FILE_

typedef struct {
  const char *mnemonic;
  const char *args;
  unsigned long image;
  unsigned long format_id;
  unsigned long pseudo_idx;
  unsigned long counter;
  unsigned long dmask;
} acasm_opcode;

typedef struct {
  const char *symbol;
  const char *cspec;
  unsigned long value;
} acasm_symbol;

extern long long getbits(unsigned int bitsize, char *location, int endian);
extern void putbits(unsigned int bitsize, char *location, long long value, int
    endian);

extern const int num_opcodes;
extern acasm_opcode opcodes[];
extern const acasm_symbol udsymbols[];
extern const int num_symbols;
extern const char *pseudo_instrs[];
extern const int num_pseudo_instrs;
```

#endif

Listing A.46: gdb/opcodes/configure.sed

```
/bfd_alpha_arch)/ i\
        bfd_xxxxx_arch)  ta="$ta xxxxx-dis.lo xxxxx-opc.lo" ;;
```

Listing A.47: gdb/opcodes/disassemble.c.sed

```
/#define ARCH_alpha/ i\
#define ARCH_xxxxx

/#ifdef ARCH_alpha/ i\
#ifdef ARCH_xxxxx \
    case bfd_arch_xxxxx: \
    disassemble = print_insn_xxxxx; \
    break; \
#endif
```

Listing A.48: gdb/opcodes/Makefile.in.sed

```
/CFILES = \\/ a\
        xxxxx-opc.c \\\
        xxxxx-dis.c \\

/ALL_MACHINES = \\/ a\
        xxxxx-opc.lo \\\
        xxxxx-dis.lo \\

/alpha-dis.lo:/ i\
xxxxx-opc.lo: xxxxx-opc.c sysdep.h config.h $(INCDIR)/ansidecl.h \\\
  $(INCDIR)/opcode/xxxxx.h \
xxxxx-dis.lo: xxxxx-dis.c sysdep.h config.h $(INCDIR)/ansidecl.h \\\
  $(INCDIR)/dis-asm.h $(BFD_H) $(INCDIR)/ansidecl.h $(INCDIR)/symcat.h \\\
  $(INCDIR)/libiberty.h $(INCDIR)/ansidecl.h $(INCDIR)/opcode/xxxxx.h \\\
  opintl.h $(BFDDIR)/elf-bfd.h $(INCDIR)/elf/common.h \\\
  $(INCDIR)/elf/internal.h $(INCDIR)/elf/external.h $(INCDIR)/bfdlink.h \\\
  $(INCDIR)/elf/xxxxx.h $(INCDIR)/elf/reloc-macros.h
```

Listing A.49: gdb/opcodes/xxxxx-dis.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* Print ___arch_name___ instructions for GDB, the GNU debugger, or for objdump.
   Copyright 2005, 2006 --- The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of GDB, GAS, and the GNU binutils.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA. */

/*
 * written by:
 * Alexandre Keunecke I. de Mendonca
 * Felipe Guimaraes Carvalho
 * Max Ruben de Oliveira Schultz
 */


#include <setjmp.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include <stdio.h>
#include "sysdep.h"
#include "dis-asm.h"
#include "opintl.h"
#include `"opcode/'___arch_name___`.h"'

/*
    ----------------------------------------------------------------
    */

#define `MAX_FORMAT_SIZE '___max_format_size___
#define `VARIABLE_FORMAT_SIZE '___variable_format_size___
#define BI_NONE 0
#define BI_MD_REL_BIT (1 << 3)
#define BI_MD_AL_BIT (1 << 6)

/*
    ----------------------------------------------------------------
    */

// Disassembler generation definitions...

/*
  Linked list with all the operands of the isntruction.
  field_id -> position of field in the instruction, later used to know about the size of
      field, (examples: '1' or '1+2')
  type -> type of field e.g. reg, addr, immm, exp
*/
typedef struct _ac_symbol {
   char *field_id;
   char *type;
   struct _ac_symbol *next;
} ac_symbol;
```

```
struct private
{
  /* Points to first byte not fetched. */
  bfd_byte * max_fetched;
  bfd_byte the_buffer[MAX_FORMAT_SIZE];
  bfd_vma insn_start;
  jmp_buf bailout;
};

/*
    ----------------------------------------------------------------
    */

/*
   bufInitial variable -- set only by parse()
   Don't ever change these variables elsewhere!
*/
char bufInitial[50];

char bufFinal[50];

/*
   Addend variables -- set only by get_addend()
   Don't ever change these variables elsewhere!
*/
static int valr = 0; // value used to return R addends
static int vala = 0; // value used to return A addends


/* assuming max input = 64 */
static int log_table[] = { 0 /*invalid*/, 0, 1, 1,
                           2 /* log 4 */, 2, 2, 2,
                           3 /* log 8 */, 3, 3, 3,
                           3, 3, 3, 3,
                           4 /* log 16 */, 4, 4, 4 };
```

```
/*
    ------------------------------------------------------------
    */

static int disassemble (bfd_vma memaddr, struct disassemble_info *info, unsigned
    long insn, int insn_size);

int 'print_insn_'___arch_name___' (bfd_vma memaddr, struct disassemble_info * info);'

static unsigned long get_field_size(unsigned long insn_fmt, int field_id);

static unsigned long get_insn_size(unsigned long insn_fmt);

ac_symbol* parse(char *args);

static unsigned int get_builtin_marker(char *s);

static void get_addend(char addend_type, char **st);

void replace(char *str, char *old, char *new);

/*
    ------------------------------------------------------------
    */


#define FETCH_DATA(info, addr) \
  ((addr) <= ((struct private *)(info->private_data))->max_fetched \
   ? 1 : fetch_data ((info), (addr)))

static int
fetch_data (struct disassemble_info *info, bfd_byte *addr)
{
```

```
  int status;
  struct private *priv = (struct private *) info->private_data;
  bfd_vma start = priv->insn_start + (priv->max_fetched - priv->the_buffer);




  status = (*info->read_memory_func) (start,
                                      priv->max_fetched,
                                      addr - priv->max_fetched,
                                      info);
  /* verify (status != 5) because for architecture with variable format size are read 8,
      16, 24 and 32 bits of memory and in the last instruction error can occur */
  if ((status != 0) && (status != 5))
    {
      (*info->memory_error_func) (status, start, info);
      longjmp (priv->bailout, 1);
    }
  else
    priv->max_fetched = addr;

  return 1;
}

static unsigned long get_field_size(unsigned long insn_fmt, int field_id)
{
___fieldsize_function___
}

static unsigned long get_insn_size(unsigned long insn_fmt)
{
___insnsize_function___
}



/*
```

```
   Function that makes the disassembler of an instruction of the object file, making the
       decoding using the field dmask of the file xxxx−opc.c

1− Using the insn apllies the mask (dmaks fiels of opcodes)
2− Search in instruction table the corresponding instruction of (insn & dmask), and
   prints the intruction mnemonic.
3− For each operand, is generated a mask in the position of the field in insn and size (e
   .g. bit 16 to the 24), after verified if is a register, or immediate, address, etc...
4− if is a register, is called the method replace to put the name of the register in the
   final string that it will be printed
5− is an immediate one!, then it verifies the modifiers, and it applies them, after calls
   the method replace to put the value in the final string(bufFinal).
6− prints string(bufFinal) in the screen with the disassembled instruction.
7− returns the octets processed, to be able to increment the address (PC)

Params
  memaddr − address of the current instruction (PC value)
  info − structure of binutils used to print the instruction and operands
  insn − the raw instruction
*/
static int disassemble (bfd_vma memaddr, struct disassemble_info *info, unsigned
     long insn, int insn_size){
   acasm_opcode *op = (acasm_opcode *) opcodes;
   acasm_symbol *symbols = (acasm_symbol *) udsymbols;
   unsigned int FORMAT_SIZE=0;

   //Find the mnemonic of instruction in opcodes table
   while (op->mnemonic){
      unsigned long e = insn & op->dmask;

      if ((e == op->image) && (op->pseudo_idx == 0) && ((unsigned)insn_size
          == get_insn_size(op->format_id))) {

         FORMAT_SIZE = get_insn_size(op->format_id);
```

```
//Print instruction mnemonic
(*info->fprintf_func) (info->stream,"%s\t",op->mnemonic);

//Now Look for Registers and address
char *args = malloc(sizeof(char)*50);
strcpy(args, op->args);
ac_symbol* acs = parse(args);

char *ptInitial = malloc(sizeof(char)*50);
char *ptFinal = malloc(sizeof(char)*50);
ptInitial = bufInitial;
ptFinal = bufFinal;
strcpy(ptFinal,ptInitial);

while (acs){
   unsigned long begining = 0;

   char *ptfield_id = malloc(sizeof(char)*50);
   ptfield_id = acs->field_id;
   unsigned long size = 0;
   unsigned int last_field_id = 0;
   while (*ptfield_id != '\0') {
      if ((*ptfield_id >= '0') && (*ptfield_id <= '9')) {
        size = size + get_field_size(op->format_id, atoi(ptfield_id));
        last_field_id = (unsigned)atoi(ptfield_id);
      }
      ptfield_id++;
   }

   unsigned int i = 0;
   for (i=0; i<=last_field_id; i++){
      begining += get_field_size(op->format_id, i);
   }

   unsigned long dmask = 0x00;
```

```
for (i=0; i<size; i++){
    dmask = (dmask << 1) + 1;
}

dmask = dmask << (FORMAT_SIZE − begining);//operand dmask
unsigned long value = dmask & insn;
value = value >> (FORMAT_SIZE − begining); //dislocates masks to see
    it which reg ex: [0,31]
int cont = 0;
int in = 0;
char symbolaux[50];



/* Search the register table, looking value (taken from the dmask) is a
    register and finds the name of the register and prints it as operating of
    instruction, if not found, will be an immediate field or another thing it
    prints the value directly,*/

while (cont < num_symbols){
    if ((strcmp(acs−>type, symbols−>cspec) == 0) && (value ==
        symbols−>value)){
        //operand find
        in = 1;
        strcpy(symbolaux, symbols−>symbol);
    }
    symbols++;
    cont++;
}
if (in){
    char *ptr2;
    ptr2 = bufFinal;
    replace(ptr2, acs−>type, symbolaux);
}else{//not operand find, it can be: exp, immm, addr
    //verifies if has modifiers and it records in vala,and valr
```

```
                unsigned int bi_type = get_builtin_marker(acs->type);

                int align = (bi_type & BI_MD_AL_BIT) ? 1 : 0;
                int pcrel = (bi_type & BI_MD_REL_BIT) ? 1 : 0;

                if (align){ //Modifier A
                    value <<= (vala ? log_table[vala] : log_table[FORMAT_SIZE / 8]);
                }
                if (pcrel){ //Modifier R
                    if (value & (1<<(size-1))){
                        value |= ((0xFFFFFFFF >> (32 - size)) & (0xFFFFFFFF << (
                            size-1)));
                        value = (((0xFFFFFFFF >> (32 - size)) & ~value) + 1);
                        value = memaddr - value + valr;
                    }
                    else
                        value += memaddr + valr;
                }
                char *ptr3;
                ptr3 = bufFinal;
                char new[50];
                //Convert long to char
                sprintf( new, "0x%01lx", value );
                replace(ptr3,acs->type,new);
            }
            symbols = symbols - cont;//Return table symbol pointer
            acs = acs->next;
        }
        break;
    }
    op++;
  }
  (*info->fprintf_func) (info->stream,"%s ",bufFinal);
  return FORMAT_SIZE / 8;
}
```

```
/*
    Function that is called by objdump core (entry point for the disassemble) for each
        instruction of the file, reads from the binary file the raw insn and calls the
        function disassemble() that disassemble the instruction.
*/
int 'print_insn_'___arch_name___'␣(bfd_vma␣memaddr,␣struct␣disassemble_info␣*␣info){'
    struct private priv;
    bfd_byte *buffer = priv.the_buffer;
    unsigned long insn = 0;
    unsigned long insnfound;
    int sizeinsn;

    info->private_data = & priv;
    priv.max_fetched = priv.the_buffer;
    priv.insn_start = memaddr;

    if (setjmp (priv.bailout) != 0)
        /* Error return. */
        return -1;

    acasm_opcode *op = (acasm_opcode *) opcodes;

    /* Verify if architecture has variable format size (CISC) ou no (RISC) */
    if (VARIABLE_FORMAT_SIZE == 0) {
        /*RISC - read fix length bits of the memory */
        FETCH_DATA (info, buffer + 4);
        insn = getbits(MAX_FORMAT_SIZE, buffer, ___endian_val___);
        insnfound = insn;
        sizeinsn = MAX_FORMAT_SIZE;
    }
    else {
        /* CISC - read variable length bits of the memory
            1- read 8 bits and find instruction in opcodes table
```

```
        2– if has format size > 8, read 16 bits and find instruction in opcodes table
        3– if has format size > 16, read 24 bits and find instruction in opcodes table
        4– if has format size > 24, read 32 bits and find instruction in opcodes table
        Decode last insn find.
      */
      unsigned int localsize = 8;
      unsigned int i;
      while (localsize <= MAX_FORMAT_SIZE) {
        FETCH_DATA (info, buffer + (localsize / 8));
        for (i=3; i>=(localsize/8); i--)
          buffer[i] = 0;
        insn = getbits(localsize, buffer, ___endian_val___);

        //Find the mnemonic of instruction in opcodes table
        int j = 0;
        while ((op->mnemonic) && (op->dmask != 0)) {
          unsigned long e = insn & op->dmask;

          if ((e == op->image) && (op->pseudo_idx == 0) && (localsize ==
              get_insn_size(op->format_id))){
            insnfound = insn;
            sizeinsn = localsize;
            break;
          }
          op++;
          j++;
        }
        localsize = localsize + 8;
        op = op - j;
      }
    }

    return disassemble(memaddr, info, insnfound, sizeinsn);
}
```

```
ac_symbol* parse(char *args){
    char *ptr_bufInitial = bufInitial;
    ac_symbol *lista = NULL;
    ac_symbol *temp;
    ac_symbol *t;
    t = NULL;
    temp = NULL;
    int i = 0;

    while (*args != '\0'){
        char buf[50];
        char *ptr_buf = buf;
        char field_aux[50];
        char *ptr_field_aux = field_aux;

        /*
          '%' is the start of a well-formed sentence. It goes like this:
          '%'<conversion specifier>':'<insn field ID>':<information from ld>:
          ' %imm:5:0:,%reg:3:0:,%reg:1:0:
        */
        if (*args == '%'){
            args++;
            while (*args != ':'){
                *ptr_buf = *args;
                *ptr_bufInitial = *args;
                ptr_bufInitial++;
                ptr_buf++;
                args++;
            }

            *ptr_buf = '\0';
            args++;

            char *arg_start = args;
```

```
while ((*args >= '0') && (*args <= '9'))
    args++;

*ptr_field_aux = *arg_start;
ptr_field_aux++;
args++;

while ((*args != ':') && (*args != '+'))
    args++;

if (*args == ':') {
  args++;
}
else if (*args == '+') {
  *ptr_field_aux = *args;
  ptr_field_aux++;
  args++;
  *ptr_field_aux = *args;
  ptr_field_aux++;
  args++;
  args++;
  while (*args != ':')
    args++;
  args++;
}

*ptr_field_aux = '\0';
ptr_field_aux++;

char *to2 = malloc(sizeof(char)*50);
strcpy(to2, field_aux);

char *to = malloc(sizeof(char)*50);
strcpy(to, buf);
```

```
            if (i > 0){
                temp = (ac_symbol *) malloc(sizeof(ac_symbol));
                temp->field_id = to2;
                temp->type = to;
                temp->next = NULL;
                lista->next = temp;
                lista = lista->next;
            }
            else{
                lista = (ac_symbol *) malloc(sizeof(ac_symbol));
                t = lista;
                lista->field_id = to2;
                lista->type = to;
                lista->next = NULL;
            }
            i++;
        }else{ //if (*args=='%')
            if (*args == '\\'){
                args++;
                args++;
                while ((*args>64 && *args<91) || (*args>96 && *args<123))
                    args++;
            }else{
                *ptr_bufInitial = *args;
                args++;
                ptr_bufInitial++;
            }
        }
    }//end of while (*args!='\0')
    *ptr_bufInitial = '\0';
    lista = t;
    return lista;
}
```

```
/*
    ------------------------------------------------------------------------
    */

/*
 Bases:
  'exp'
        'addr'
        'imm'

 Modifiers:
  'H'[n][c][u|s] -> n -> get the n-high bit -- u or s (unsigned or unsigned) -- c -
      add carry from lower bits
  'L'[n][u|s] -> get the n-low bits
        'R'[n][b] -> backward, n -> add n to PC
        'A'[n][u|s] -> align in a paragraph of n-bits

 st : null-terminated string with the conversion specifier
*/
static unsigned int get_builtin_marker(char *st){
    valr = 0;
    vala = 0;

    unsigned int ret_val = BI_NONE;

    if (*st == 'e' && *(st+1) == 'x' && *(st+2) == 'p'){
        st += 3;
    }
    else if (*st == 'a' && *(st+1) == 'd' && *(st+2) == 'd' && *(st+3) == 'r'){
        st += 4;
    }
    else if (*st == 'i' && *(st+1) == 'm' && *(st+2) == 'm'){
        st += 3;
    }
    else
```

```
        return BI_NONE;

    while (*st != '\0'){
        switch (*st){
            case 'A':
                ret_val |= BI_MD_AL_BIT;
                get_addend('A', &st);
                break;
            case 'R':
                ret_val |= BI_MD_REL_BIT;
                get_addend('R', &st);
               break;
        }
        st++;
    }

    return ret_val;
}

/*
    ----------------------------------------------------------------
    */

/*
   Parse and get the modifier's addend
   Only called by get_builtin_marker()

   addend_type: ''A' or 'R'

   st : string pointing to the addend type. It will be set to the next char
   after the addend
*/
static void get_addend(char addend_type, char **st){
    char *sl = *st;
    sl++;
```

```
    while (*sl >= '0' && *sl <= '9')
        sl++;

    if ((sl-1) != *st){
        char savec = *sl;
        *sl = '\0';

        if (addend_type == 'A')
            vala = atoi((*st)+1);
        else // 'R'
            valr = atoi((*st)+1);

        *sl = savec;
    }

    switch (addend_type){
        case 'R': // [b]
            if (*sl == 'b'){
              sl++;
              valr = valr * (-1);
            }
            break;
        case 'A': // [u|s]
            if (*sl == 's'){
              sl++;
            }
            else if (*sl == 'u'){
              sl++;
            }
            break;
    }

    *st = sl-1;
}
```

```
/*
    Function that from parameter 'ptr_ret' search for the first occurrence of 'old' and
        substitutes for 'new'
    e.g.: ptr_ret = addi reg,reg,exp
        old = reg
        new = $30
    result:
        ptr_ret = addi $30,reg,exp
*/
void replace(char *ptr_ret, char *old, char *new){
    char a[50];
    char *str = a;

    strcpy(str, ptr_ret);

    while (*str != '\0'){
        if (strncmp(old, str, strlen(old)) == 0){
            while(*old == *str){
                old++;
                if (*str != '\0')
                    str++;
            }
            while (*new != '\0'){
                *ptr_ret = *new;
                ptr_ret++;
                new++;
            }
            while (*str != '\0'){
                *ptr_ret = *str;
                str++;
                ptr_ret++;
            }
        }
```

```
        else{
            *ptr_ret = *str;
            str++;
            ptr_ret++;
        }
    }

    *ptr_ret = '\0';
}
```

Listing A.50: gdb/opcodes/xxxxx-opc.c

```
/* ex: set tabstop=2 expandtab:
   -*- Mode: C; tab-width: 2; indent-tabs-mode nil -*-
*/
/* ___arch_name___'-opc.c' -- ___arch_name___ opcode list.
   Copyright 2005, 2006 --- The ArchC Team

This file is automatically retargeted by ArchC binutils generation tool.
This file is part of GDB, GAS, and the GNU binutils.

GDB, GAS, and the GNU binutils are free software; you can redistribute
them and/or modify them under the terms of the GNU General Public
License as published by the Free Software Foundation; either version
1, or (at your option) any later version.

GDB, GAS, and the GNU binutils are distributed in the hope that they
will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See
the GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this file; see the file COPYING. If not, write to the Free
Software Foundation, 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA. */

/*
```

```
 * written by:
 * Alexandro Baldassin
 */

#include <stdio.h>
#include "sysdep.h"
#include '"opcode/'___arch_name___'.h"'

acasm_opcode opcodes[] = {
___opcode_table___
};

const acasm_symbol udsymbols[] = {
___symbol_table___
};

const char *pseudo_instrs[] = {
___pseudo_table___
};

/*
 * 1 = big, 0 = little
 *
 * limitations:
 * . endianess affects 8-bit bytes
 * . maximum word size: 64-bit
 */
long long getbits(unsigned int bitsize, char *location, int endian)
{
  long long data = 0;
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;

  if (bits_remain) number_bytes++;
```

```
  int index, inc;
  if (endian == 1) { /* big */
    index = 0;
    inc = 1;
  }
  else { /* little */
    index = number_bytes - 1;
    inc = -1;
  }

  while (number_bytes) {
    data = (data << 8) | (unsigned char)location[index];

    index += inc;
    number_bytes--;
  }

/*
 * if bitsize is not multiple of 8 then clear/pack the remaining bits
 */
  long long mask = 0;
  for (; bitsize; bitsize--)
    mask = (mask << 1) | 1;

  if (bits_remain) {
    if (endian == 1) {
      long long temp = data >> (8 - bits_remain);
      temp &= (mask << bits_remain);

      data = temp | (data & ~(mask << bits_remain));
    }
  }

  return data & mask;
}
```

```
void putbits(unsigned int bitsize, char *location, long long value, int endian)
{
  unsigned int number_bytes = (bitsize / 8);
  unsigned int bits_remain = bitsize % 8;
  unsigned char bytes[8];
  unsigned int i;
  int index;
  unsigned char mask = 0;

  /*
        * Fill the bytes array so as not to depend on the host endian
        *
        * bytes[0] -> least significant byte
        */
  for (i=0; i<8; i++)
    bytes[i] = (value & (0xFF << (i*8))) >> 8*i;

  if (bits_remain) {
    for (i=0; i<bits_remain; i++)
          mask = (mask << 1) | 1;
  }

  if (endian == 0) { /* little */
    index = 0;

    while ((unsigned int)index != number_bytes) {
      location[index] = bytes[index];

      index++;
    }

        if (bits_remain)
```

```
      location[number_bytes] = (location[number_bytes] & ~mask) | (bytes[number_bytes
          ] & mask);

  }
  else { /* big */
    index = number_bytes - 1;

    i = 0;
    while (index >= 0) {
      if (bits_remain)
        location[i] = ((bytes[index+1] & mask) << (8-bits_remain)) | (bytes[index]
            >> bits_remain);
      else
        location[i] = bytes[index];

      i++;
      index--;
    }

    if (bits_remain)
      location[number_bytes] = (bytes[0] & mask) | (location[number_bytes] & ~mask);
  }
}

const int num_opcodes = ((sizeof opcodes) / (sizeof(opcodes[0])));
const int num_symbols = ((sizeof udsymbols) / (sizeof(udsymbols[0])));
const int num_pseudo_instrs = ((sizeof pseudo_instrs) / (sizeof(pseudo_instrs[0])));
```