Exercício Programa 1 - Minips

Arquitetura de Computadores - 2021.Q1

Emilio Francesquini - e.francesquini@ufabc.edu.br

15 de fevereiro de 2021

1 O Projeto

Neste projeto vamos implementar a primeira fase de um emulador para a ISA MIPS de 32 bits. Como é um MIPS reduzido, recebeu o carinhoso apelido de MINIPS. 😎 Este emulador vai ser capaz de rodar diversos pequenos programas que acompanham este enunciado. Mais detalhes sobre isto abaixo.

Para tirar a nota máxima neste primeiro EP, seu programa deve executar corretamente todos os programas de entrada disponibilizados. Como o conjunto de instruções usado por estes programas é reduzido, vocês efetivamente terão que implementar apenas algumas poucas instruções no seu emulador.

Recomendo que os seguintes passos sejam seguidos para fazer este EP:

- Escolha a linguagem de sua preferência. Contudo, escolha uma linguagem não muito exotérica para ter a certeza que a linguagem não vai se tornar a parte difícil do trabalho.
- Escreva o código que simula a memória (pode ser muito bem uma tabela de hash com chaves sendo os endereços e os valores a palavra armazenada).
- Escreva o código que carrega a seção de dados e de instruções na memória (mais detalhe sobre as convenções abaixo).
- Escreva o código que decodifica o código binário em um estrutura de dados que você consiga usar facilmente no seu programa.
- Implemente a primeira parte deste EP que é imprimir o assembly do programa dado como entrada.

- Implemente o emulador, instrução a instrução, conforme elas forem aparecendo nos programas de teste. Lembre-se: é um MINIPS, não o MIPS completo, então tudo bem se, por enquanto, ele só rodar esses poucos problemas de exemplo.

2 Convenções

- Apesar do livro dizer que o MIPS é big-endian. Ele na verdade vem nos dois sabores: big e little. **Os arquivos de programas de teste fornecidos são little-endian.**

- Para facilitar a vida não vamos usar arquivos ELF que são o formato de arquivo binário executável no Linux. Eu já separei a seção de dados e de texto em arquivos binários distintos para facilitar. Os nomes dos arquivos sempre seguem o formato `nn.nome.secao`, onde `nn` é o número do programa de entrada (quanto maior o número mais complexo), `nome` descreve o que o programa faz, e `secao` pode ser ou `text` ou `data` representando as seções de texto (binário executável) ou de dados (variáveis globais e estáticas).

- Todos os registradores e posições de memória (exceto os mencionados explicitamente abaixo) devem ser inicializados com `0` no início da simulação.

- Seguindo a convenção de outros emuladores (veja mais na seção recursos), nós usaremos os seguintes endereços na inicialização:

Descrição	**Endereço (hexa)**
Início da seção de texto	`0x00400000`
Início da seção de dados	`0x10010000`
`$sp`	`0x7fffeffc`
`$gp`	`0x10008000`
Program Counter (PC)	Mesmo da seção de texto (`0x00400000`)

3 Syscalls

Normalmente a entrada e saída de periféricos é feita através do mapeamento em memória de seus registradores. Aqui, para simplificar, vamos usar a mesma convenção dos emuladores SPIM e MARS.

Uma *syscall* é uma instrução especial que simula uma chamada ao sistema operacional. O código binário completo de uma syscall é sempre: `0x0000000c` (no seu decodificador talvez seja mais fácil assumir que uma syscall é uma instrução do tipo R com o `funct` igual a `0x0c`). Os valores de registradores são utilizados para determinar o que a chamada de sistema fará. Você precisará implementar as seguintes chamadas de sistema para poder executar todos os programas de exemplo dados.

Valor de `$v0`	Descrição da Syscall
1	Imprime o inteiro contido em `$a0`
4	Imprime String (terminada por `\0`) apontada pelo endereço em `$a0`
5	Lê um inteiro e o coloca em `$v0`
10	Termina a execução do programa
11	Imprime o caractere em `$a0`

Exemplo de código (o binário em hexa está ao lado de cada instrução):

```
# 5 é o código para ler um inteiro
addi $v0, $zero, 5  # 0x20020005
syscall             # 0x0000000c

# Coloca o valor lido em $a0
add $a0, $zero, $v0 # 0x00022020
# 1 é o código para imprimir um inteiro
addi $v0, $zero, 1  # 0x20020001
#Imprime inteiro lido
syscall             # 0x0000000c

# \n é o caractere 10
addi $a0, $zero, 10 # 0x2004000a
# 11 é o código para imprimir um caractere
addi $v0, $zero, 11 # 0x2002000b
# Imprime uma quebra de linha
syscall             # 0x0000000c

#Finaliza a execução
addi $v0, $zero, 10 # 0x2402000a
syscall             # 0x0000000c
```

4 Recursos e cuidados

Você pode e deve usar os recursos que estiverem disponíveis na internet para implementar o seu EP. Contudo, você não pode usar códigos prontos de outras pessoas. Trechos de código do StackOverflow, por exemplo, podem ser usados tranquilamente contanto que a fonte seja citada com um comentário no código fonte.

Para comparar o seu programa com uma referência, o professor implementou essa primeira fase. Os binários podem ser encontrados aqui:

- Linux (x86-64): minips
- Windows (x86-64): minips.exe

Você também pode usar um emulador de MIPS para rodar o código passo a passo. Eu recomendo usar o MARS (`http://courses.missouristate.edu/kenvollmar/mars/`) que é gratuito e tem o mesmo comportamento esperado para este EP (inclusive das syscalls).

Este site `https://godbolt.org/` permite você escrever funções simples e gerar o assembly para MIPS 32 (selecione MIPS GCC 5.4 e como opções ao lado coloque `-O0` para evitar as otimizações do compilador e acabar com um código incompreensível). Pode ser uma boa ideia utilizá-lo em conjunto com o MARS (que é capaz de exportar o formato binário do texto e dos dados usados como entrada deste projeto) para testar alguns programas simples que você escrever.

Durante a implementação tenha cuidado, em especial, com a representação binária das instruções e dados. Lembre-se, estamos trabalhando em little-endian. Tome cuidado com extensão de sinal (quando deve-se quando não se deve aplicar) isso faz muita diferença dependendo da instrução (veja exemplo da `slt` nos slides da aula).

Além do capítulo do livro, existe uma folha de referência do MIPS que pode ser encontrada aqui: Greencard.

5 Entradas e saídas esperadas

O seu programa deve se comportar como o programa de exemplo fornecido oferecendo duas opções de execução. Uma que decodifica o binário e imprime o assembly e a outra que efetivamente executa o código.

Abaixo algumas saídas do programa fornecido:

```
$ ./minips decode 02.hello
```

```
LUI $at, 4097
ORI $a0, $at, 0
ADDIU $v0, $zero, 4
SYSCALL
ADDIU $a0, $zero, 10
ADDIU $v0, $zero, 11
SYSCALL
ADDIU $v0, $zero, 10
SYSCALL
$ ./minips run 02.hello
Ola mundo!
Execution finished successfully
-------------------------
Instruction count: 9 (R: 3 I: 6 J: 0)
IPS: 9429.899697300219s
$
```

Em caso de dúvidas, envie para o Discord da disciplina.

As entradas de exemplo podem ser baixadas aqui (entradas.zip) e contêm:

Entrada	Descrição
`01.soma`	Soma 3 + 4 e imprime o resultado
`02.hello`	Hello World!
`03.input`	Lê um inteiro e imprime na saída
`04.branches`	Testes simples com instruções de branches
`05.fibo`	Imprime a sequência de Fibonacci
`06.collatz`	Conta quantos elementos tem a sequência de Collatz
`07.loadstore`	Testes simples com instruções de load/store
`08.sort`	QuickSort em um vetor de elementos

Note que o programa de referência lê automaticamente os arquivos com extensão `.text` e `.data` dado o prefixo do teste desejado.

6 Entrega

Junto com o seu código você deve entregar um brevíssimo relatório (máximo de 2 páginas) contendo obrigatoriamente:

- Nome completo

- RA
- Usuário do GitHub
- Link para vídeo no YouTube ou qualquer outro lugar acessível pelo professor.

O seu vídeo deve ter duração máxima de 5 minutos, deve mostrar o seu código executando e deve destacar os pontos fortes da sua implementação assim como as eventuais limitações e dificuldades que você teve durante a implementação.

A entrega do código e do relatório deve ser feita pelo GitHub Classroom através do link https://classroom.github.com/a/JU4EkkID. Será considerado como entrega o último *commit* (não esqueça de dar *push*) no repositório até a **data limite de 14/03/2021**.

Projetos entregues com atraso sofrerão descontos seguindo a seguinte tabela:

Dias em atraso	Nota máxima
1 dia	7
2 dias	6
3 dias	5
>3 dias	0

Para discussões, dúvidas e comentários utilize o Discord em https://discord.gg/9RtRcx3.