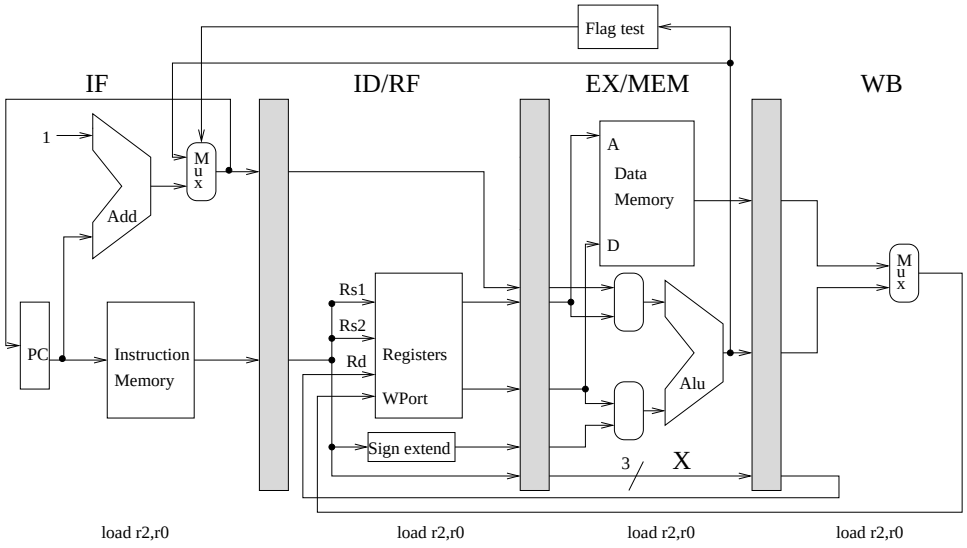
Arquitecturas Avançadas de Computadores (AAC)
2009/10



Guia de Laboratório I

Simulação ao nível funcional do processador μRISC

Demonstração: 30 e 31/3/2010, nas aulas de laboratório

Departamento de Engenharia Electrotécnica e de Computadores
Instituto Superior Técnico

Fevereiro de 2010

1 Introdução

Este trabalho visa a realização de um simulador funcional, (em C, C++, VHDL ou Java) do processador μRISC, de 16 bits com arquitectura RISC. Esse processador possui 8 registos de uso geral e 42 instruções. Cada instrução executada no μRISC demora quatro ciclos para completar. Esses ciclos são denominados: IF, ID, EX e WB. No primeiro ciclo (IF), a próxima instrução a ser executada é carregada de memória e armazenada no registo de instruções (IR). No segundo ciclo (ID), a instrução é descodificada e os operandos são lidos do banco de registos. No terceiro ciclo (EX), a instrução é executada e são calculadas as condições dos resultados (condition codes). No quarto ciclo (WB), os resultados são escritos no banco de registos.

2 Implementação do Processador μRISC

O processador μRISC tem as seguintes características:

- 16 bits;
- 8 registos de uso geral de 16 bits de largura (r0 ... r7);
- 42 instruções;
- instruções de 3 operandos;
- organização de dados na memória do tipo *big endian*;
- memória endereçada ao nível de palavra, i.e., cada endereço de memória corresponde a uma palavra de 2 bytes. O processador tem 16 bits de endereço. A capacidade total de memória do processador é de 32KB (16k endereços x 2 bytes).

Para se entender como deve programar o simulador funcional do processador μRISC é fundamental compreender o significado do termo ”funcional”. Uma descrição funcional divide o processador nos principais blocos que o constituem (implementação). Portanto, o processador deverá conter quatro rotinas principais, uma para cada ciclo a ser executado em cada instrução. Note que a única forma de comunicação entre essas quatro rotinas num processador real é através de registos, que no caso do simulador devem ser representados por estruturas de dados no programa de simulação. Além dessas quatro rotinas, os elementos principais do processador real deverão estar encapsulados por módulos, tais como os elementos, banco de registos, ALUs, incrementadores e memória.

2.1 Ambiente de simulação

Independentemente da linguagem utilizada, o simulador deverá:

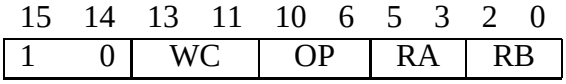
- Ler um ficheiro de texto com um conjunto de instruções a ser executado, codificados em hexadecimal.
- Gerar uma listagem do estado do processador e dos primeiros K octetos de memória no fim da execução do programa, assim como o número de ciclos gasto.

2.2 Conjunto de Instruções

O conjunto de instruções do processador μRISC pode ser classificado em 4 grandes grupos:

- instruções aritméticas que envolvem a ALU;
- instruções que envolvem o uso de constantes;
- instruções de transferência de controlo;
- instruções de acesso a memória.

2.2.1 ALU



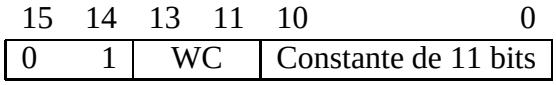
OP	Operação	Mnemónica	*Flags* actualizadas
00000	$C = A + B$	add c, a, b	todas
00001	$C = A + B + 1$	addinc c, a, b	todas
00011	$C = A + 1$	inca c, a	todas
00100	$C = A - B - 1$	subdec c, a, b	todas
00101	$C = A - B$	sub c, a, b	todas
00110	$C = A - 1$	deca c, a	todas
01000	C = Shift Lógico Esq. (A)	lsl c, a	S,C,Z
01001	C = Shift Aritmético Dir. (A)	asr c, a	S,C,Z
10000	$C = 0$	zeros c	nenhuma
10001	$C = A\&B$	and c, a, b	Z,S
10010	$C =!A\&B$	andnota c, a, b	Z,S
10011	$C = B$	passb c, b	nenhuma
10100	$C = A\&!B$	andnotb c, a, b	Z,S
10101	$C = A$	passa c, a	Z,S
10110	$C = A \oplus B$	xor c, a, b	Z,S
10111	$C = A\|B$	or c, a, b	Z,S
11000	$C =!A\&!B$	nor c, a, b	Z,S
11001	$C =!(A \oplus B)$	xnor c, a, b	Z,S
11010	$C =!A$	passnota c, a	Z,S
11011	$C =!A\|B$	ornota c, a, b	Z,S
11100	$C =!B$	passnotb c, b	Z,S
11101	$C = A\|!B$	ornotb c, a, b	Z,S
11110	$C =!A\|!B$	nand c, a, b	Z,S
11111	$C = 1$	ones c	nenhuma

2.2.2 Constantes

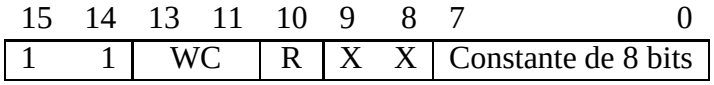
O processador μRISC possui somente instruções para o carregamento de constantes com sinal, representadas pelas instruções a seguir indicadas:

Formato	Operação	Mnemónica
I	C = Constante	loadlit c, Const
II	C = Const8 $\mid$ $(C \& 0xff00)$	lcl c, Const8
II	C = (Const8 $<<$ 8) $\mid$ $(C \& 0x00ff)$	lch c, Const8

Formato I:



Formato II:

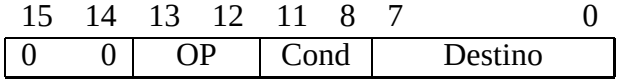


R	Operação
0	lcl
1	lch

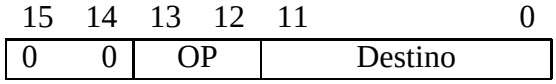
2.2.3 Transferência de controlo

O processador μRISC possui cinco instruções para a transferência de controlo, codificadas das três maneiras diferentes abaixo indicadas. A primeira codificação é utilizada para instruções de salto condicional (onde o deslocamento de 8 bits especificado é relativo ao PC) e a segunda codificação é utilizada para a instrução de salto incondicional (onde o deslocamento de 12 bits é também relativo ao PC). A terceira codificação é reservada para saltos indirectos por registos, onde o endereço especificado no registo é o endereço absoluto.

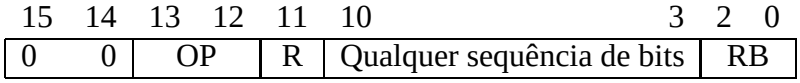
Formato I:



Formato II:



Formato III:



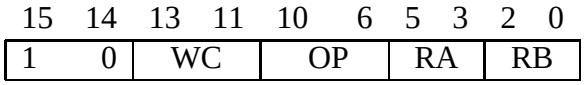
Formato	OP	Operação	Mnemónica
I	00	Jump False	jf.cond Destino
I	01	Jump True	jt.cond Destino
II	10	Jump Incondicional	j Destino
III	11	Jump and Link	jal b
III	11	Jump Register	jr b

No caso da instrução de *jump and link*, o próximo **PC** (PC+1) deve ser armazenado no registo **r7** e o conteúdo do registo **RB** armazenado em **PC**. Já no caso da instrução *jump register*, a única operação a ser feita é armazenar o conteúdo do registo **RB** no registo registo **PC**.

R	Operação
0	Jump and Link
1	Jump Register

COND	Condição	Mnemônica
0100	Resultado ALU negativo	neg
0101	Resultado ALU zero	zero
0110	Carry da ALU	carry
0111	Resultado ALU negativo ou zero	negzero
0000	TRUE	true
0011	Resultado ALU overflow	overflow

2.2.4 Memória



OP	Operação	Mnemônica
01010	C = Mem[A]	load c, a
01011	Mem[A] = B	store a, b

2.2.5 NOP

Diversas instruções podem gerar NOPs. No caso, foi escolhida a condição **jf.true 0x00**, ou seja, uma instrução onde todos os 16 bits estão a zero.

2.2.6 Fim da execução

A instrução **HALT** (paragem do sistema) deve ser implementada como **L: j L**, isto é, um salto incondicional para o presente endereço.

3 Formato do ficheiro de programa

Esta secção define o formato do ficheiro de instruções para o processador μRISC, a ser adoptado, obrigatoriamente, por todos os grupos. Conforme especificado, a memória do processador possui 16 bits de endereçamento, sendo endereçada palavra a palavra. Ou seja, pode-se endereçar até 65536 palavras de 16 bits cada (os endereços vão de 0 a 65535 em decimal ou de 0000 a 0xffff em hexadecimal), mas dispõe-se apenas de 32k *bytes* de memória.

3.1 Formato do ficheiro de programa

O formato do ficheiro de instruções para o processador μRISC é simples, tendo como características principais:

- O ficheiro, em modo texto, deve conter apenas uma instrução por linha.
- As instruções devem estar codificadas em hexadecimal.

- A palavra-chave **address** é usada para definir a posição de memória a partir da qual as instruções são ser armazenadas: **address end**, onde **end** é um endereço de memória codificado em hexadecimal.

Como exemplo ilustrativo da utilidade desta palavra-chave, considere o seguinte ficheiro de instruções:

```
address   00ff
77ff
c041
address   1000
c868
2fff
```

Neste caso, as instruções **77ff** e **c041** serão armazenadas nas posições de memória **00ff(hex)** e **0100(hex)**, respectivamente, e as instruções **c868** e **2fff** nas posições de memória **1000(hex)** e **1001(hex)**, respectivamente. Se a directiva **address** não for usada para definir a localização das instruções, estas deverão ser colocadas a partir da posição de memória **0000**.

3.2 Exemplo de um ficheiro de instruções

Apresenta-se, a seguir, um ficheiro de instruções que obedece ao formato acima descrito.

```
.module teste
.pseg
main:
     ; r0 points to the stack
     loadlit r6,-1
     lcl r0, LOWBYTE X
     lch r0, HIGHBYTE X
     jal r0
HLT: j HLT
X:   loadlit r4,29
     loadlit r3,0x0F
     add  r2,r3,r4
     jr   r7
.dseg
ARRAY:
.end
```

```
address    0000
77ff
c005
c400
3000
2fff
601d
580f
901c
3807
address    0009
```

O conteúdo de memória após o processador carregar o ficheiro de instruções acima é:

Endereço de memória	Conteúdo
0x0000 (Posição 0)	77ff
0x0001 (Posição 1)	c005
0x0002 (Posição 2)	c400
0x0003 (Posição 3)	3000
0x0004 (Posição 4)	2fff
0x0005 (Posição 5)	601d
0x0006 (Posição 6)	580f
0x0007 (Posição 7)	901c
0x0008 (Posição 8)	3807

Agradecimentos

As especificações do processador μRISC foram criadas pelo Prof. Claudionor Coelho, da Universidade Federal de Minas Gerais em Belo Horizonte.

Referências

[1] J. L. Henessy e D. A. Patterson, Computer Architecture, A Quantitative Approach, 3^a edição, Morgan Kaufmann Publishers, Inc., 2003.
[2] D. A. Patterson e J. L. Hennessy. Computer Organization & Design - The Hardware/Software Interface. Morgan Kaufmann Publishers, Inc., 1994.