Universidade do Minho

Módulo 5

A organização sequencial do Y86

1. *Introdução*

No final deste módulo os alunos deverão ser capazes de:

- descrever a organização sequencial do Y86;
- enumerar e instanciar com valores concretos os sinais de controlo relevantes para a execução de cada instrução do Y86
- acompanhar passo a passo a execução de uma sequências de instruções usando o simulador **ssim**.

1.1. Conteúdos e Resultados de Aprendizagem relacionados

Conteúdos	9.1 – *Datapath* Sequencial
Resultados de Aprendizagem	R9.1 – Analisar e descrever organizações sequenciais de processadores elementares

2. *Material de apoio*

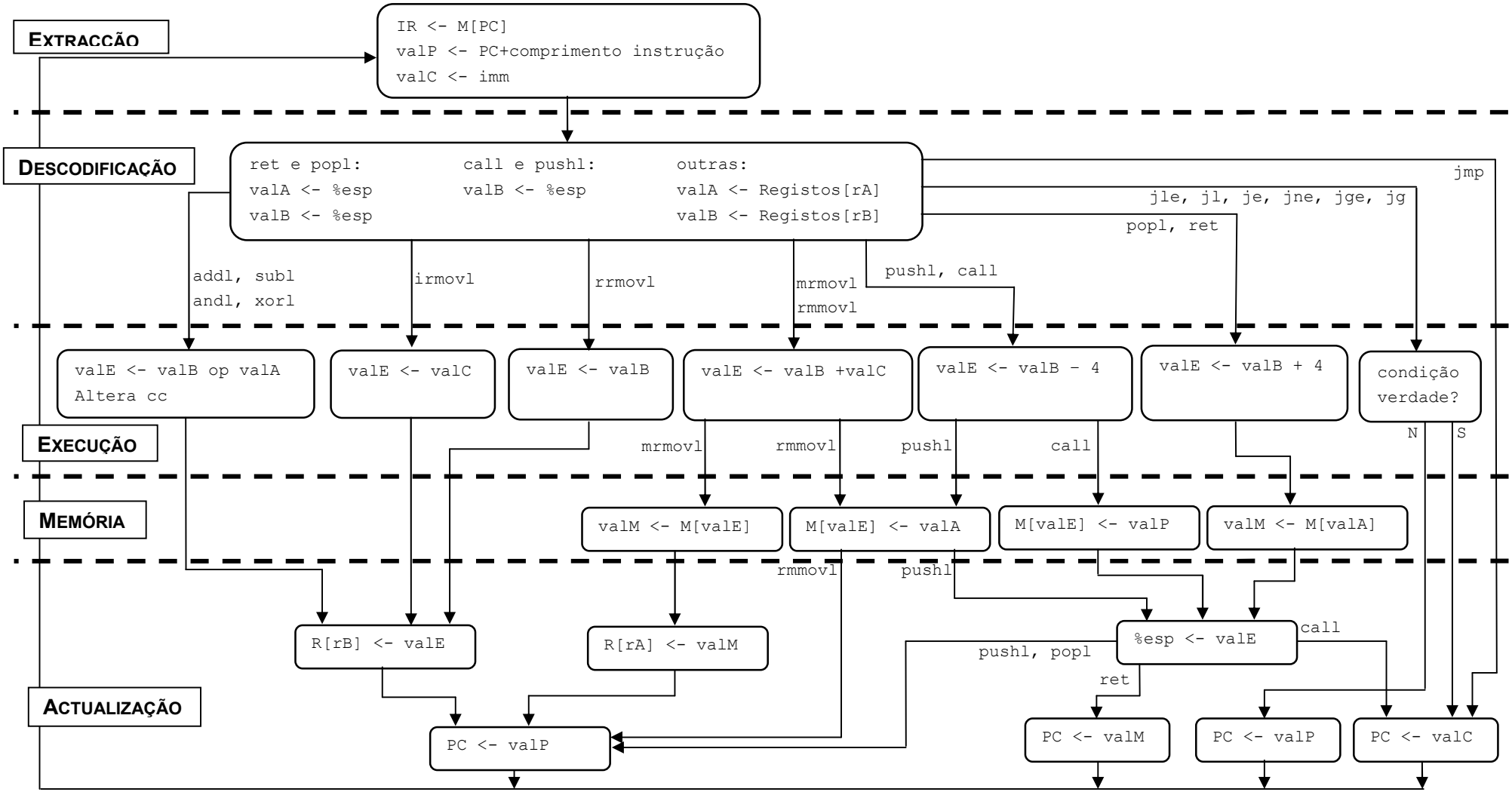
A bibliografia relevante para este módulo é a secção 4.3 do livro "Computer Systems: a Programmer's Perspective", de Randal E. Bryant e David O'Hallaron.

Em anexo a este módulo encontra-se um diagrama de blocos desta organização (http://alba.di.uminho.pt/lei/ac/).

O conjunto de ferramentas de apoio a este módulo, **ssim**, pode ser carregado a partir do web site do livro, no endereço http://csapp.cs.cmu.edu/public/students.html

Estão disponíveis as ferramentas em formato binário e também o código-fonte. Esta instalação necessita do tcl/tk. Se não os tiver correctamente instalados na sua máquina, pode encontrá-los em http://tcl.sourceforge.net/.

3. *Fluxograma arquitectura SEQ Y86*



4. Estágios de execução de uma instrução

Para realizar este exercício carregue da página da disciplina o ficheiro soma.ya que contem o *assembly* Y86 de invocação e definição de uma função de soma de dois inteiros (e que a seguir se reproduz).

soma.ya

```
# Execução começa no endereço 0
  .pos 0
init:
  irmovl Stack, %esp      # ***
  irmovl Stack, %ebp
  jmp main                 # ***

main:
  irmovl $4, %eax
  pushl %eax
  irmovl $24, %eax         # ***
  pushl %eax               # ***
  call Soma                # ***
  halt
Soma:
  pushl %ebp
  rrmovl %esp, %ebp        # ***
  mrmovl $8(%ebp), %eax
  mrmovl $12(%ebp), %ebx   # ***
  addl %ebx, %eax          # ***
  rrmovl %ebp, %esp
  popl %ebp                # ***
  ret                       # ***

  .pos 0x100
Stack:  # Inicio da pilha
```

Para cada instrução assinalada com *** preencha uma tabela com os valores dos sinais de controlo da organização sequencial. Apresente os sinais diferenciados pelo estágio em que são relevantes/gerados. Para cada tipo de instrução que surja pela primeira vez apresente os seus valores genéricos, conforme o exemplo que se segue (correspondente às 2 primeiras instruções assinaladas do programa). A sua tarefa poderá ser mais simples se gerar o ficheiro objecto (soma.yo) com o **yas**. Verifique as suas respostas usando o simulador[1] "**ssim –g soma.yo"**.

Estágio	**Genérico**	**Específico**
	irmovl V, %esp	irmovl Stack, %esp
Extracção	icode:ifun = M1[PC] rA:rB = M1[PC+1] valC = M4[PC+2] valP = PC+ 6	icode:ifun = M1[000]= 3:0 rA:rB = M1[001] = 8:4 valC = M4[002] = 0x0100 valP = 000 + 6 = 6

[1] Para executar o simulador em modo gráfico deve copiar a *script* `seq.tcl` de `/usr/local/bin/` para a sua directoria de trabalho.

Descodificação		
Execução	valE = valC+0	valE = 0x0100
Memória		
Actualização	R[rB] = valE	%esp = 0x0100
PC	PC = valP	PC = 6

Estágio	**Genérico**	**Específico**
	jmp Dest	jmp main
Extracção	icode:ifun = M1[PC] valC = M4[PC+1] valP = PC+5	icode:ifun = M1[0x00c]= 7:0 valC = M4[0x00d] = 0x0011 valP = 0x00c + 5 = 0x011
Descodificação		
Execução		
Memória		
Actualização		
PC	PC = valC	PC = 0x011