

 Universidade do Minho	Arquitectura de Computadores II L.E.S.I. – 3º ano	
	Módulo nº 2 A organização sequencial do Y86	

1. Introdução

Pretende-se com este módulo:

- apresentar a organização sequencial do Y86;
- introduzir os estágios lógicos de execução de uma instrução genérica;
- apresentar os sinais de controlo desta organização e a forma como se relacionam com a execução de cada instrução.

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`.

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://gec.di.uminho.pt/lesi/ac2/>).

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. 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.

Para cada instrução assinalada com *** preencha uma tabela com os valores dos sinais de controlo da organização sequencial. Apresente os sinais diferenciados pelo estágio de execução 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 com o `yas`. Verifique as suas respostas usando o simulador¹ "`ssim -g soma.yo`".

Estágio	Genérico	Específico
		<code>irmovl V, %esp</code>
Extracção	<code>icode:ifun = M1[PC]</code> <code>rA:rB = M1[PC+1]</code> <code>valC = M4[PC+2]</code> <code>valP = PC+ 6</code>	<code>icode:ifun = M1[000]= 3:0</code> <code>rA:rB = M1[001] = 8:4</code> <code>valC = M4[002] = 0x0100</code> <code>valP = 000 + 6 = 6</code>
Descodificação		
Execução	<code>valE = valC+0</code>	<code>valE = 0x0100</code>
Memória		
Actualização	<code>R[rB] = valE</code>	<code>%esp = 0x0100</code>
PC	<code>PC = valP</code>	<code>PC = 6</code>

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

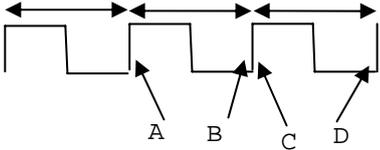
4. Temporização

Considere a seguinte sequência de código:

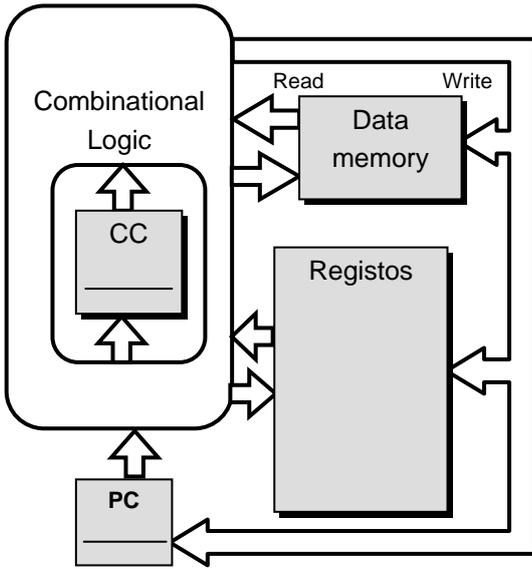
Ciclo	Endereço	Instrução
1	0x0000	<code>irmovl \$100, %eax</code>
2	0x0006	<code>subl %eax, %eax</code>
3	0x0008	<code>je loop</code>

Preencha as figuras abaixo, com os valores dos registos, PC e códigos de condição, para os instantes A, B, C e D assinalados no seguinte diagrama temporal:

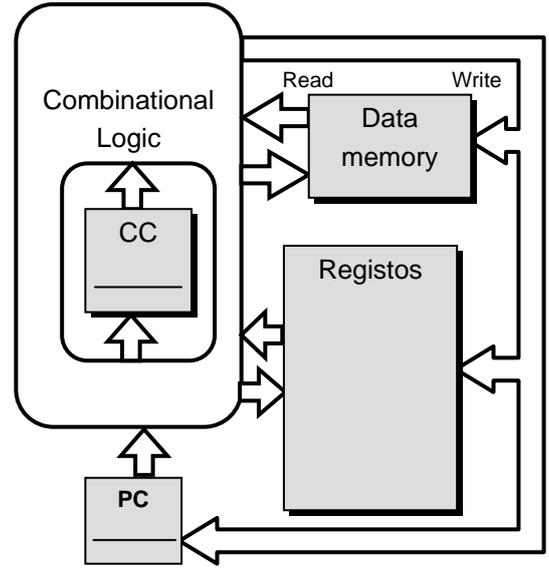
¹ Para executar o simulador em modo gráfico deve copiar a `script seq.tcl` de `/usr/local/bin/` para a sua directoria de trabalho.



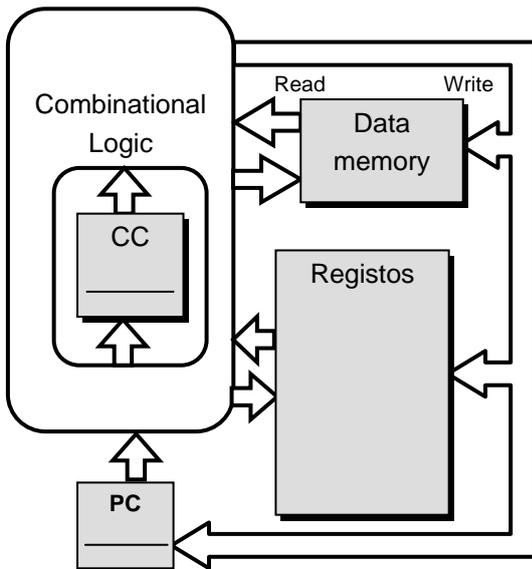
Instante A



Instante B



Instante C



Instante D

