Universidade do Minho

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

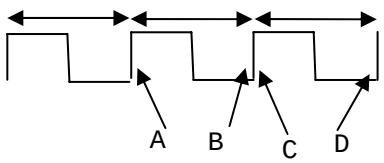
Estágio	Genérico	Específico
	`jmp Dest`	`jmp main`
Extracção	icode:ifun = M1[PC] valC = M4[PC+1] valP = PC+5	icode:ifun = M1[0x00c]= 7:0 valC = M4[0x00d] = 0x0011 valP = 0x00c + 5 = 0x011
Descodificação		
Execução		
Memória		
Actualização		
PC	PC = valC	PC = 0x011

4. Temporização

Considere a seguinte sequência de código:

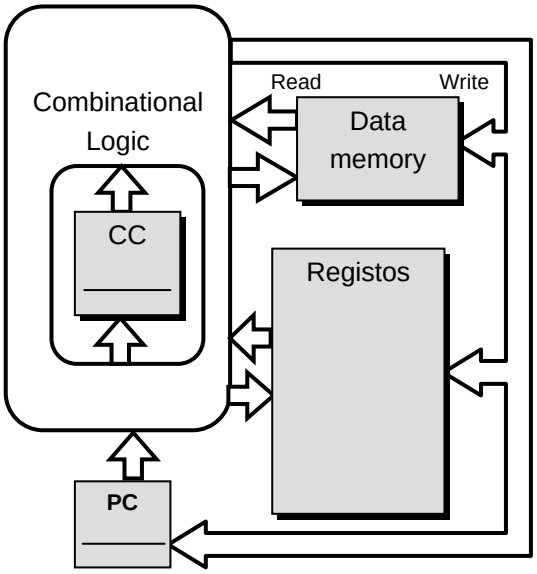
Ciclo	Endereço	Instrução
1	0x0000	irmovl $100, %eax
2	0x0006	subl %eax, %eax
3	0x0008	je loop

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:

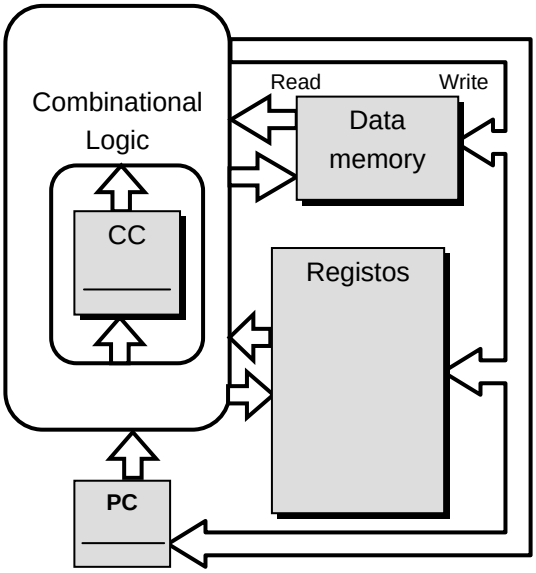


[1] Para executar o simulador em modo gráfico deve copiar a *script* `seq.tcl` de `/usr/local/bin/` para a sua directoria de trabalho.

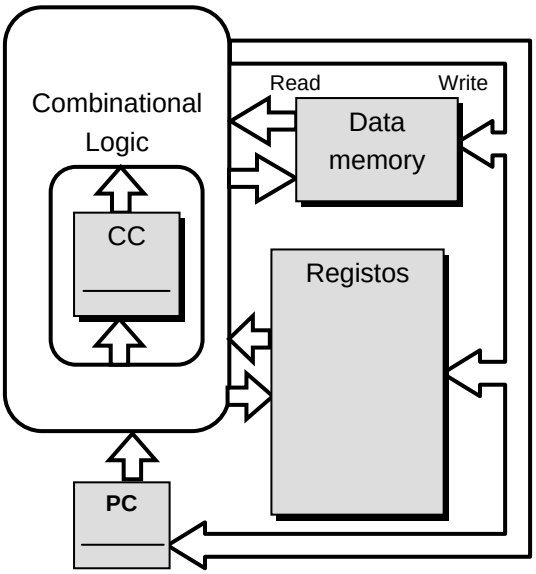
Instante A



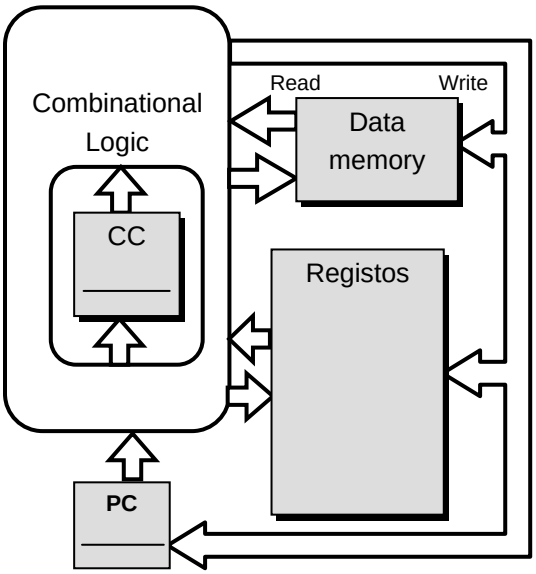
Instante B



Instante C



Instante D



5. Extensão ao conjunto de instruções

Para cada uma das instruções (ou tipo de instruções) apresentadas nas próximas alíneas indique:

1. se a sua realização é possível usando o *data path* SEQ do Y86;
2. caso afirmativo:
   i. como poderia esta instrução ser codificada (icode,ifun, rA,rB, valC);
   ii. quais os valores dos vários sinais do *datapath* (à semelhança da secção 3)
3. caso negativo proponha alterações ao *datapath* que permitiriam suportar essa instrução.

a) Operações lógicas e aritméticas com operando imediato

O Y86 apenas suporta operações lógicas e aritméticas em que ambos os operandos são registos. Estude a possibilidade de suportar operações com um operando imediato (`addil, subil, andil, xoril`), com o seguinte comportamento:

OP imm, rB -> rB = rB OP imm

b) Saltos com endereçamento em registo

O Y86 apenas suporta saltos absolutos com o endereço destino indicado como um valor imediato. Estude a possibilidade de suportar saltos absolutos com o destino do salto indicado como um registo:

jxx rA -> PC = R[rA]

c) leave

O IA32 inclui esta instrução que realiza 2 operações:

```
movl %ebp, %esp
popl %ebp
```

Estude a possibilidade de suportar o `leave` no Y86.