

PROVA SCRITTA DEL MODULO DI
CALCOLATORI ELETTRONICI
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA
CORSO DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA E INFORMATICA
 18 Settembre 2018

NOME:

COGNOME:

MATRICOLA:

ESERCIZIO 1 (10 punti)

- (6 punti) Progettare una rete logica che calcoli il complemento a 2 di un operando a tre bit.
- (4 punti) Indicare a quale tipo di reti logiche appartiene la rete progettata e spiegarne il motivo.

ESERCIZIO 2 (8 punti)

Si consideri un calcolatore che dispone di una memoria principale di 256 Mbyte e di una memoria cache di 512 Kbyte. E' possibile accedere al singolo byte e la memoria è suddivisa in blocchi da 16 byte.

- (4 punti) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache nel caso venga usata la modalità di indirizzamento
 - Diretto
 - Completamente associativo
- (2 punti) Si considerino le due parole di indirizzo rispettivamente 21A0x43 e 10y0F4D. Calcolare i valori delle cifre x e y tali per cui le due parole si trovino nella stessa "linea" di cache nel caso di indirizzamento diretto.
- (2 punti) Nel caso di indirizzamento associativo, ipotizzando che la cache sia piena e il blocco in cui trova la prima parola di cui al punto precedente sia già in cache, mentre non lo sia il blocco in cui si trova la seconda, l'eventuale chiamata a questa determinerà una sostituzione del blocco della prima? Motivare la risposta, facendo riferimento agli algoritmi di sostituzione FIFO e LRU.

ESERCIZIO 3 (8 punti)

- (4 punti) Scrivere la sequenza di istruzioni MIPS che implementino lo pseudo-codice:

$$A[i] \leftarrow A[i] + 1$$

Dove A è un indirizzo simbolico di memoria e i rappresenta l'offset rispetto all'indirizzo dato. Nell'implementare la soluzione, ricordare i vincoli di distanza tra una parola e la successiva nel MIPS.

- (4 punti) Interpretare il seguente frammento di codice Assembly MIPS, istruzione dopo istruzione, considerando e modificando lo stato della memoria e dei registri a partire da quello in figura (PC è il contatore di programma).

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|------|-------------------|------|-------------------|--|--|------|--------|-----|----|--|--|-----|-----|--|------|----|--------|------|----|--|--|------|---|--|--|------|----|
| Circoscrittore di programma). | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <pre>inizio: muli \$4, \$4, 4 addi \$5, \$0, 1000 jr \$5 exit: ...</pre> | <table><tr><td>\$4</td><td>2</td><td>1000</td><td>sw \$8, 1024(\$4)</td></tr><tr><td></td><td></td><td>1004</td><td>j exit</td></tr><tr><td>\$5</td><td>-5</td><td></td><td></td></tr><tr><td>\$8</td><td>150</td><td></td><td>....</td></tr><tr><td>PC</td><td>inizio</td><td>1024</td><td>15</td></tr><tr><td></td><td></td><td>1028</td><td>7</td></tr><tr><td></td><td></td><td>1032</td><td>-1</td></tr></table> | \$4 | 2 | 1000 | sw \$8, 1024(\$4) | | | 1004 | j exit | \$5 | -5 | | | \$8 | 150 | | | PC | inizio | 1024 | 15 | | | 1028 | 7 | | | 1032 | -1 |
| \$4 | 2 | 1000 | sw \$8, 1024(\$4) | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1004 | j exit | | | | | | | | | | | | | | | | | | | | | | | | | | |
| \$5 | -5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| \$8 | 150 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PC | inizio | 1024 | 15 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1028 | 7 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1032 | -1 | | | | | | | | | | | | | | | | | | | | | | | | | | |

ESERCIZIO 4 (7 punti)

Si consideri il caso di un Bus di I/O sul quale devono essere collegate 8 periferiche esterne.

Si ipotizzi che sulla parte di controllo del Bus si abbiano solo 2 segnali di controllo liberi da poter utilizzare per gestire l'arbitraggio delle 8 periferiche da collegare. Descrivere la tecnica di arbitraggio utilizzabile in una tale situazione, disegnando un possibile schema di collegamento delle periferiche e spiegando chiaramente le implicazioni del suo utilizzo sulla gestione delle richieste di I/O delle periferiche.

ESERCIZIO 1

Tabella di verità della rete in oggetto :

| Ingressi | | | Uscite | | |
|----------|---|---|--------|----|----|
| A | B | C | A' | B' | C' |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

| AB \ C | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| 0 | | 1 | | 1 |
| 1 | 1 | 1 | | |

$$A' = \bar{A}B + \bar{A}C + A\bar{B} \cdot \bar{C}$$

| AB \ C | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| 0 | | 1 | 1 | |
| 1 | 1 | | | 1 |

$$B' = B\bar{C} + \bar{B}C$$

| AB \ C | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| 0 | | | | |
| 1 | 1 | 1 | 1 | 1 |

$$C' = C$$

Si tratta ovviamente di una rete logica combinatoria, in quanto le uscite dipendono soltanto dai valori degli ingressi ad un dato istante.

ESERCIZIO 2

1. Per indirizzare 256 Mbyte occorre un indirizzo di almeno 28 bit. Per indirizzare il singolo byte all'interno di un blocco occorrono 4 bit ($16 = 2^4$), che coincidono con i 4 bit meno significativi dell'indirizzo di memoria primaria. I restanti 24 bit costituiscono l'indirizzo del "block frame".

(a) Indirizzamento diretto. In questo caso devo poter indirizzare ciascuno dei 32K blocchi contenuti nella cache ($512\text{Kbyte}/(16\text{byte}/\text{blocco})$). Occorrono 15 bit che coincidono con i 15 bit meno significativi del "block frame" Pertanto i 28 bit di indirizzo della memoria primaria vengono interpretati come:

| tag | cache index | offset |
|-------|-------------|--------|
| 9 bit | 15 bit | 4 bit |

(b) Indirizzamento (completamente) associativo. Non è presente il campo "cache index". I 28 bit di indirizzo della memoria primaria vengono interpretati come:

| tag (block frame) | offset |
|-------------------|--------|
| 24 bit | 4 bit |

2. Perché le due parole si trovino nella stessa "linea" di cache è necessario eguagliare l'index delle due parole. Faremo riferimento alla suddivisione in campi dell'indirizzo mostrato nella soluzione della prima parte dell'esercizio.

Nel caso di indirizzamento diretto avremo:

| tag | cache index | Offset |
|-----------|-----------------|--------|
| 9 bit | 15 bit | 4 bit |
| 001000011 | 0100000xxxx0100 | 0011 |
| 00010000y | yyy000011110100 | 1101 |

In esadecimale, $x = F$, mentre y può essere pari a 2 od A perché la cifra più significativa è nel tag.

3. La sostituzione del blocco contenente la prima parola con il blocco contenente la seconda dipende essenzialmente dal fatto che, a seconda dell'algoritmo FIFO/LRU, quel blocco si presenti come candidato alla sostituzione. Nell'indirizzamento associativo, il cache index o il tag non hanno alcun ruolo in questa decisione. In particolare:

- Algoritmo FIFO: il blocco verrà sostituito se si troverà in testa alla coda di sostituzione (il "primo" chiamato in ordine di tempo).
- Algoritmo LRU: il blocco verrà sostituito se risulterà essere il meno referenziato all'istante della chiamata della parola in esame.

In tutti gli altri casi, ad essere sostituito sarà un altro blocco.

ESERCIZIO 3

Soluzione domanda 1.

Supponendo che `i` sia memorizzato nel registro `$1`, le istruzioni prevedono:

- Calcolo dell'offset completo
- Istruzione di trasferimento da memoria a registro
- Aggiornamento del registro
- Aggiornamento della memoria con trasferimento da registro

```
mul $1, $1, 4
lw $4, A($1)
addi $4, $4, 1
sw $4, A($1)
```

Soluzione domanda 2.

In neretto l'istruzione in esecuzione e il relativo effetto sulla memoria e/o registri.

| | |
|--|---|
| inizio: <code>mul \$4, \$4, 4</code> <code>addi \$5, \$0, 1000</code> <code>jr \$5</code> exit: <code>...</code> | <div> <div> <div>\$4</div> <div>8</div> </div> <div> <div>\$5</div> <div>-5</div> </div> <div> <div>\$8</div> <div>150</div> </div> <div> <div>PC</div> <div>inizio+4</div> </div> </div> <div> <div>1000</div> <div>1004</div> <div>1024</div> <div>1028</div> <div>1032</div> </div> <div> <div>Memoria</div> <div>sw \$8, 1024(\$4)</div> <div>j exit</div> <div>....</div> <div>15</div> <div>7</div> <div>-1</div> </div> |
| inizio: <code>mul \$4, \$4, 4</code> <code>addi \$5, \$0, 1000</code> <code>jr \$5</code> exit: <code>...</code> | <div> <div> <div>\$4</div> <div>8</div> </div> <div> <div>\$5</div> <div>1000</div> </div> <div> <div>\$8</div> <div>150</div> </div> <div> <div>PC</div> <div>inizio+8</div> </div> </div> <div> <div>1000</div> <div>1004</div> <div>1024</div> <div>1028</div> <div>1032</div> </div> <div> <div>Memoria</div> <div>sw \$8, 1024(\$4)</div> <div>j exit</div> <div>....</div> <div>15</div> <div>7</div> <div>-1</div> </div> |
| inizio: <code>mul \$4, \$4, 4</code> <code>addi \$5, \$0, 1000</code> <code>jr \$5</code> exit: <code>...</code> | <div> <div> <div>\$4</div> <div>8</div> </div> <div> <div>\$5</div> <div>1000</div> </div> <div> <div>\$8</div> <div>150</div> </div> <div> <div>PC</div> <div>1000</div> </div> </div> <div> <div>1000</div> <div>1004</div> <div>1024</div> <div>1028</div> <div>1032</div> </div> <div> <div>Memoria</div> <div>sw \$8, 1024(\$4)</div> <div>j exit</div> <div>....</div> <div>15</div> <div>7</div> <div>-1</div> </div> |

| | |
|--|---|
| <pre>inizio: muli \$4, \$4, 4 addi \$5, \$0, 1000 jr \$5 exit: ...</pre> | <div><div><div>\$4</div><div>8</div></div><div><div>\$5</div><div>1000</div></div><div><div>\$8</div><div>150</div></div><div><div>PC</div><div>1004</div></div></div> <div><div>1000</div><div>sw \$8, 1024(\$4)</div></div> <div><div>1004</div><div>j exit</div></div> <div><div>1024</div><div>15</div></div> <div><div>1028</div><div>7</div></div> <div><div>1032</div><div>150</div></div> |
| <pre>inizio: muli \$4, \$4, 4 addi \$5, \$0, 1000 jr \$5 exit: ...</pre> | <div><div><div>\$4</div><div>8</div></div><div><div>\$5</div><div>1000</div></div><div><div>\$8</div><div>150</div></div><div><div>PC</div><div>exit</div></div></div> <div><div>1000</div><div>sw \$8, 1024(\$4)</div></div> <div><div>1004</div><div>j exit</div></div> <div><div>1024</div><div>15</div></div> <div><div>1028</div><div>7</div></div> <div><div>1032</div><div>150</div></div> |
| <pre>inizio: muli \$4, \$4, 4 addi \$5, \$0, 1000 jr \$5 exit: ...</pre> | <div><div><div>\$4</div><div>8</div></div><div><div>\$5</div><div>1000</div></div><div><div>\$8</div><div>150</div></div><div><div>PC</div><div>exit+4</div></div></div> <div><div>1000</div><div>sw \$8, 1024(\$4)</div></div> <div><div>1004</div><div>j exit</div></div> <div><div>1024</div><div>15</div></div> <div><div>1028</div><div>7</div></div> <div><div>1032</div><div>150</div></div> |

ESERCIZIO 4

Con soli 2 segnali liberi è possibile solo utilizzare la tecnica "daisy chain" che ha solo 2 segnali (request e grant). Ovviamente questa tecnica implica una gestione rigida delle priorità, che sono definite una volta per tutte.

