

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI  
CALCOLATORI ELETTRONICI  
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**

21 Giugno 2005

**MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI**

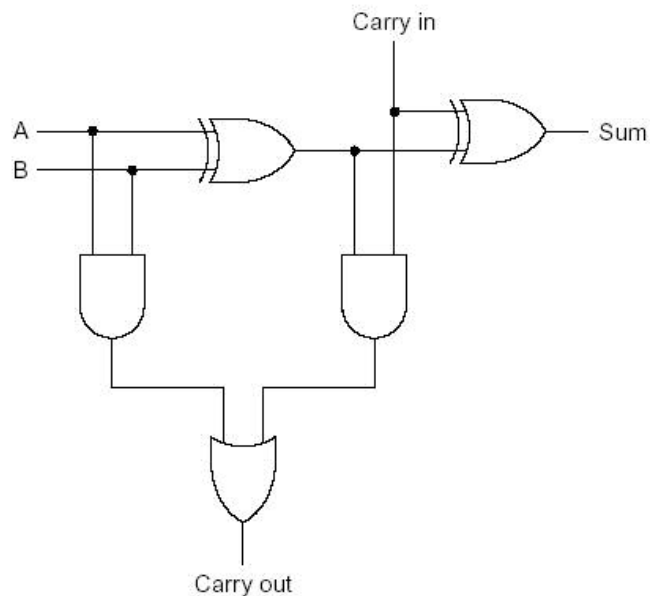
**ESERCIZIO 1 (NO: 8 punti – VO: 7 punti)**

- (a) (NO: 4 punti – VO: 3 punti) Si scriva la tavola di verità di un Full Adder e il relativo circuito impiegando il numero minimo di porte logiche.
- (b) (4 punti) Si definisca una rete logica realizzabile con N Full Adder, con relative caratteristiche e schema logico.

**Soluzione.**

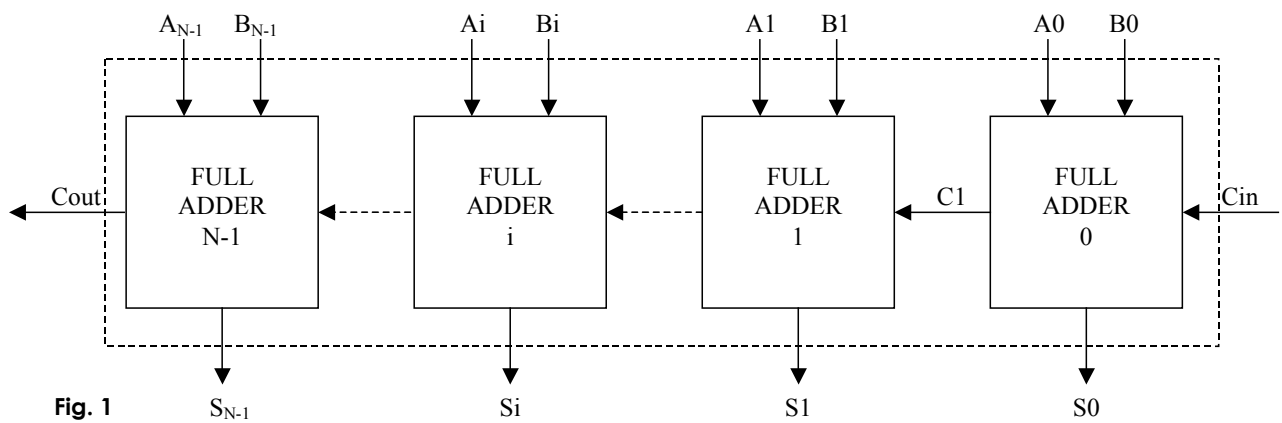
(a)

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

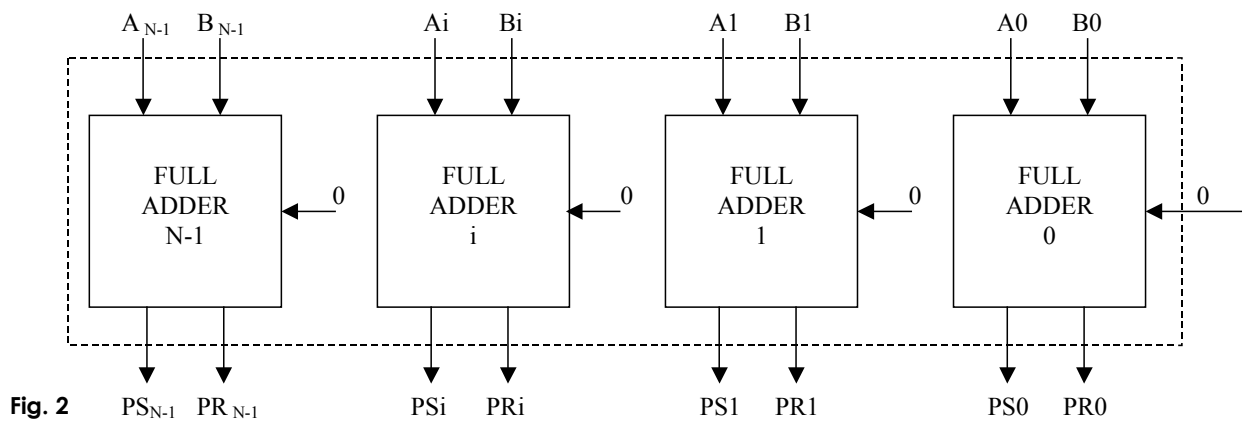


(b)

Con N Full Adder è possibile realizzare un Parallel Adder. Se infatti si hanno due parole di N bit da sommare, i bit i-esimi di ciascuna parola sono gli ingressi del Full Adder i-esimo ( $i=0, \dots, N-1$ , col bit in posizione 0 nella funzione di bit meno significativo). Il riporto in uscita del Full Adder i-esimo va collegato al riporto in ingresso del Full Adder (i+1)-esimo. Il bit di riporto in ingresso del Parallel Adder corrisponde così al bit in ingresso del Full Adder relativo ai bit meno significativi. Lo schema progettuale è il seguente:



Un'altra rete implementabile è il Carry Save-Adder, costituito da N Full Adder in parallelo:



PS e PR non sono somma e riporto dei due bit, bensì soltanto Pseudo-Somma e Pseudo-Riporto. Per ottenere la somma complessiva, bisogna moltiplicare per 2 lo Pseudo-Riporto (shift) e poi effettuare la somma  $PS + 2PR$  con un Parallel Adder.

## ESERCIZIO 2 (8 punti)

Si consideri una gerarchia di memoria cache-primaria, con una memoria cache di 8 parole ed una memoria primaria di 128 parole. Il metodo di indirizzamento usato è il metodo diretto con blocchi di due parole.

- 1) (NO: 2 punti – VO: 1 punto) Indicare, specificando il significato e la funzione dei diversi campi, come viene recuperata l'informazione nella cache a partire dall'indirizzo della parola in memoria primaria.
- 2) (NO: 6 punti – VO: 4 punti) Indicare lo stato finale della memoria cache e calcolare l'hit ratio in funzione della seguente sequenza di chiamate (in decimale, prima parola indirizzo 0): 62, 71, 58, 70, 63, 65, 59, 63, 60.
- 3) (solo VO: 3 punti) Sapendo che la cache presenta un tempo medio di accesso pari a 4 ns, si calcoli il massimo tempo medio di accesso alla primaria che consente di ottenere un tempo medio di accesso alla gerarchia inferiore od uguale a 16 ns.

### Soluzione.

1)

Indirizzamento: 7 bit

< Tag 4 bit > < Cache Index 2 bit > < Offset 1 bit >

2)

	0	1	2	3
0	64	58	60	62
1	65	59	61	63

Sequenza delle chiamate e relativo hit ratio:

Chiam.	62	71	58	70	63	65	59	63	60
C.I.	3	3	1	3	3	0	1	3	2
Hit?				Sì			Sì	Sì	

Hit ratio (H): 3/9

3)

$T_c = 4 \text{ ns}$ .

$T = T_c + (1-H) * T_p$  (formula del tempo medio di accesso alla gerarchia)

$T_c + (1-H) * T_p \leq 16 \text{ ns}$  (vincolo progettuale)

$$4 + (1-3/9) * T_p \leq 16$$

$$4 + 6/9 * T_p \leq 16$$

$$12 + 2 * T_p \leq 48$$

$$2 * T_p \leq 36$$

$$T_p \leq 18 \text{ ns}$$

Il massimo tempo medio di accesso alla primaria per rispettare le specifiche è dunque pari a 18 ns.

**ESERCIZIO 3 (solo NO: 9 punti)**

1) (3 punti) Tradurre in Assembler MIPS la seguente espressione C:

$$B[i] = B[i] * C[i];$$

Si consideri che:  $\&B[0] \rightarrow$  indirizzo 128;  $\&C[0] \rightarrow$  indirizzo 0;  $i \rightarrow \$8$ . Si utilizzi il numero MINIMO di registri necessario.

- 2) (3 punti) Si supponga che all'interno di una funzione  $f$  avvenga la chiamata ad una procedura  $g$ . I parametri in ingresso di  $f$  sono memorizzati nei registri  $\$4$  ed  $\$5$ , ed altrettanto deve avvenire per la procedura  $g$ . Si ipotizzi che  $f$  calcoli e memorizzi i parametri necessari a  $g$  nei registri  $\$22$  e  $\$23$ . Illustrare quali istruzioni devono essere presenti nella funzione  $f$  per realizzare correttamente la chiamata ed il ritorno dalla procedura  $g$ .
- 3) (3 punti) Tradurre in Assembler MIPS la seguente espressione C:

```
if (a<b)
    m = b;
else
    m = a;
```

Si consideri che:  $a \rightarrow \$4$ ,  $b \rightarrow \$5$ ,  $m \rightarrow \$6$ .

**Soluzione.**

1)

$$\$8 \leftarrow i * 4; \$9 \leftarrow B[i]; \$10 \leftarrow C[i]$$

```
mul $8, $8, 4
lw $9, 128($8)
lw $10, 0($8)
mul $9, $9, $10
sw $9, 128($8)
```

2)

Nel punto di salvataggio del contesto, la funzione  $f$  deve contenere l'istruzione:

```
sw $31, X($29)
```

per il salvataggio del registro  $\$31$ , che contiene il PC per il ritorno alla funzione chiamante la  $f$ .  
Dualmente, nel punto di ripristino del contesto:

```
lw $31, X($29)
```

$X$  è la posizione nello stack dove si vuole sia memorizzato il contenuto di  $\$31$ .

Inoltre, poiché la procedura  $g$  fa uso degli stessi registri di  $f$ , la struttura del codice dovrà essere grosso modo la seguente:

```
move $20, $4 #salvataggio dei parametri di f
move $21, $5
move $4, $22 #preparazione dei registri prima della chiamata a g
move $5, $23
jal g
move $4, $20 #ripristino dei parametri di f
move $5, $21
```

dove abbiamo usato i registri  $\$20$  e  $\$21$  per copie temporanee dei parametri di  $f$ . Un'alternativa è il salvataggio di  $\$4$  e  $\$5$  nello stack nella fase di salvataggio del contesto. Anch'essi andranno ripristinati prima di uscire dalla funzione  $f$ . Essenziale per il funzionamento di  $g$  è che i parametri che essa richiede siano presenti in  $\$4$  e  $\$5$  **prima** che essa venga chiamata con `jal`.

3)

```
slt $8, $4, $5
beq $8, $0, else
move $6, $5
j exit
else: move $6, $4
exit: ...
```

**ESERCIZIO 3 (solo VO: 6 punti)**

Si supponga di disporre di tre macchine: a pila, a uno e a due indirizzi. Per ognuna di queste si abbiano le seguenti istruzioni:

A pila		A un indirizzo		A due indirizzi	
Istruzione	Semantica	Istruzione	Semantica	Istruzione	Semantica
PUSH X	$M[X] \rightarrow \text{push}$	STORE X	$\text{ACC} \rightarrow M[X]$	MOV X1, X2	$M[X1] \rightarrow M[X2]$
POP X	$\text{pop} \rightarrow M[X]$	LOAD X	$M[X] \rightarrow \text{ACC}$	ADD X1, X2	$M[X1] + M[X2] \rightarrow M[X2]$
ADD	$\text{pop} + \text{pop} \rightarrow \text{push}$	ADD X	$\text{ACC} + M[X] \rightarrow \text{ACC}$	DIV X1, X2	$M[X1] / M[X2] \rightarrow M[X2]$
DIV	$\text{pop1} / \text{pop2} \rightarrow \text{push}$	DIV X	$\text{ACC} / M[X] \rightarrow \text{ACC}$		

ACC è il registro accumulatore della macchina a un indirizzo.

$M\{X\}$  indica il dato nella locazione di memoria X.

Si scriva, per ognuna delle tre macchine, la sequenza delle istruzioni necessarie per realizzare la seguente operazione:

$$Z = (A + B) / (C + A).$$

Le lettere indicano le locazioni di memoria dove si trovano i dati.

Nella macchina a due indirizzi si faccia uso di una ulteriore locazione P dove memorizzare i risultati parziali in modo da non perdere i dati iniziali contenuti nelle locazioni A, B o C.

**Soluzione.**

Zero indirizzi	Un indirizzo	Due indirizzi
PUSH A	LOAD A	MOV A, Z
PUSH C	ADD C	ADD C, Z
ADD	STORE Z	MOV A, P
PUSH A	LOAD A	ADD B, P
PUSH B	ADD B	DIV P, Z
ADD	DIV Z	
DIV	STORE Z	
POP Z		

**ESERCIZIO 4 (NO: 8 punti – VO: 7 punti)**

a) (2 punti) Spiegare in modo chiaro e sintetico le differenze tra memoria paginata e memoria segmentata.

b) (NO: 6 punti – VO: 5 punti) Si consideri un sistema operativo con gestione della memoria segmentata e paginata. Ciascun job viene suddiviso al massimo in 3 segmenti. Ciascun segmento ha una dimensione massima di 128 pagine. Le pagine hanno dimensione 1KB. Si abbia ad esempio la seguente situazione:

Tabella dei Segmenti	
Segmento	Tabella delle Pagine
00	2
01	0
10	1

Tabella delle Pagine 0			
Pagina Virtuale	Pagina Fisica	Bit di Validità	Solo lettura
0000000	0001010	1	0
0000001	0010001	0	0
0000010	1011001	1	0
0000011	1011010	1	0
0000100	0011101	1	0
0000101	0101111	0	0
0000110	0110111	1	0
0000111	0100000	1	0
0001000	0100100	1	0
0001001	0001001	1	0

Tabella delle Pagine 1			
Pagina Virtuale	Pagina Fisica	Bit di Validità	Solo lettura
0000000	0000011	0	0
0000001	0010110	0	0
0000010	1001001	1	0
0000011	1001010	1	0
0000100	1010101	1	0
0000101	0011101	1	0
0000110	0111111	1	0
0000111	1011101	1	0
0001000	1010011	1	0
0001001	0001111	1	0
0001010	0011011	1	0
0001011	0100010	1	0

Tabella delle Pagine 2			
Pagina Virtuale	Pagina Fisica	Bit di Validità	Solo lettura
0000000	0100001	1	1
0000001	0101110	1	1
0000010	0110110	0	1
0000011	0000110	1	1
0000100	1100011	0	1
0000101	1000011	1	1
0000110	0010101	1	1

Il bit di validità indica se la pagina virtuale richiesta è presente (1) o no (0) nella memoria principale. Si considerino le seguenti richieste e mostrare: a quale indirizzo fisico corrispondono gli indirizzi virtuali, se l'operazione può essere conclusa con successo o se viene generato un errore (segmento non valido, pagina non valida, violazione di protezione) o un page fault:

- 1) Read 000000010101001101
- 2) Write 1000001000101011001
- 2) Read 0100001011001110011
- 3) Write 0000000111100100110
- 4) Read 0100010000001100001
- 5) Write 1110000100000100111

**Soluzione:**

a) V. dispense del corso.

b)

- 1) Indirizzo fisico valido 01011100101001101
- 2) Indirizzo fisico valido 10101010101011001
- 3) Indirizzo fisico non valido (pagina non valida)
- 4) Violazione di scrittura sull'indirizzo fisico 00001101100100110
- 5) Indirizzo fisico valido 01001000001100001
- 6) Segmento non valido

**ESERCIZIO 5 (solo VO: 5 punti)**

Descrivere in modo chiaro e sintetico la classificazione di Flynn delle architetture parallele, spiegando che tipo di calcolatore ricade in ciascuna tipologia di architettura.

**Soluzione.**

Vedi le dispense del corso.