

**SECONDA PROVA INTERMEDIA DEL CORSO DI**  
**CALCOLATORI ELETTRONICI**  
**NUOVO ORDINAMENTO DIDATTICO**  
10 Giugno 2009

**NOME:**

**COGNOME:**

**MATRICOLA:**

**ESERCIZIO 1 (8 punti)**

1. (4 punti) Progettare una ALU che esegua le seguenti operazioni su due operandi (A e B) a N bit, rappresentati in complemento a 2, utilizzando un parallel adder e le opportune reti logiche: A+B, -A, A-1, A-B (rispettivamente per i valori dei bit di selezione S<sub>0</sub> e S<sub>1</sub> pari a 00, 01, 10, 11). Gli ingressi della ALU, oltre i 2 operandi, comprendono i 2 bit di selezione che indicano l'operazione da eseguire (N.B.: non c'è nessun bit di riporto in ingresso alla ALU, ma internamente esiste il bit di riporto in ingresso al parallel adder). E' richiesta anche la rappresentazione dello schema circuitale.
2. (2 punti) Proporre uno schema architetturale differente, se possibile, per *massimizzare* la velocità di calcolo della ALU.
3. (2 punti) Se A e B fossero rappresentati come interi senza segno, a cosa corrisponderebbero le operazioni eseguite dalla ALU appena implementata?

**ESERCIZIO 2 (8 punti)**

Si consideri un calcolatore in cui la CPU esegue 10<sup>5</sup> istruzioni/s. L'esecuzione di una istruzione richiede 5 cicli di clock, 3 dei quali tengono occupato il bus di sistema. L'80% dell'Instruction Rate è usato dalla CPU per eseguire programmi che non contengono trasferimenti di I/O. L'ampiezza della linea dati del bus è pari a 32 bit. Nell'ipotesi che il trasferimento di una parola di 32 bit richieda un ciclo di clock, calcolare la massima velocità di trasferimento (in KB/s):

1. (2 punti) in modalità block transfer DMA (blocchi di una parola);
2. (2 punti) in modalità transparent DMA;
3. (2 punti) in modalità DMA con furto di ciclo, in cui il modulo DMA ruba 2 cicli/istruzione.
4. (2 punti) Si discutano sinteticamente le diverse modalità di indirizzamento delle periferiche, illustrandone vantaggi e svantaggi.

**ESERCIZIO 3 (8 punti)**

Si scriva una *procedura* Assembly MIPS che esegua il prodotto scalare tra 2 vettori, v e w (di uguale dimensione N), passati per indirizzo rispettivamente in \$4 e \$5. La dimensione N dei vettori viene passata alla procedura usando \$6. Si ritorni il valore del prodotto scalare in \$7. La procedura deve occuparsi del salvataggio e del ripristino del contesto, secondo la modalità "callee save". Si ricorda che il prodotto scalare di 2 vettori si calcola come

$\sum_{i=1}^N v_i w_i$ , dove l'indice i rappresenta l'i-esimo elemento di ciascun vettore.

**ESERCIZIO 4 (8 punti)**

Si consideri il seguente insieme di processi:

Processo	Tempo di arrivo	Tempo di CPU richiesto	Memoria richiesta
1	0.00	1.40	50K
2	0.50	1.00	30K
3	1.00	0.70	20K
4	1.30	0.10	10K
5	2.95	0.30	20K

la memoria disponibile (100K), è gestita dal sistema operativo mediante partizionamento dinamico. All'istante iniziale, si consideri la memoria vuota. I processi sono gestiti con strategia FIFO multiprogrammata "round robin".

1. (4 punti) Mostrare graficamente lo sviluppo dei processi e lo stato della memoria.
2. (2 punti) Calcolare il tempo di turnaround medio e di turnaround pesato medio.
3. (2 punti) Si spieghi brevemente il concetto di virtualizzazione della memoria e la sua utilità. Si discuta inoltre cosa avviene quando si verifica un "page fault", conseguentemente alla richiesta di una determinata pagina virtuale.

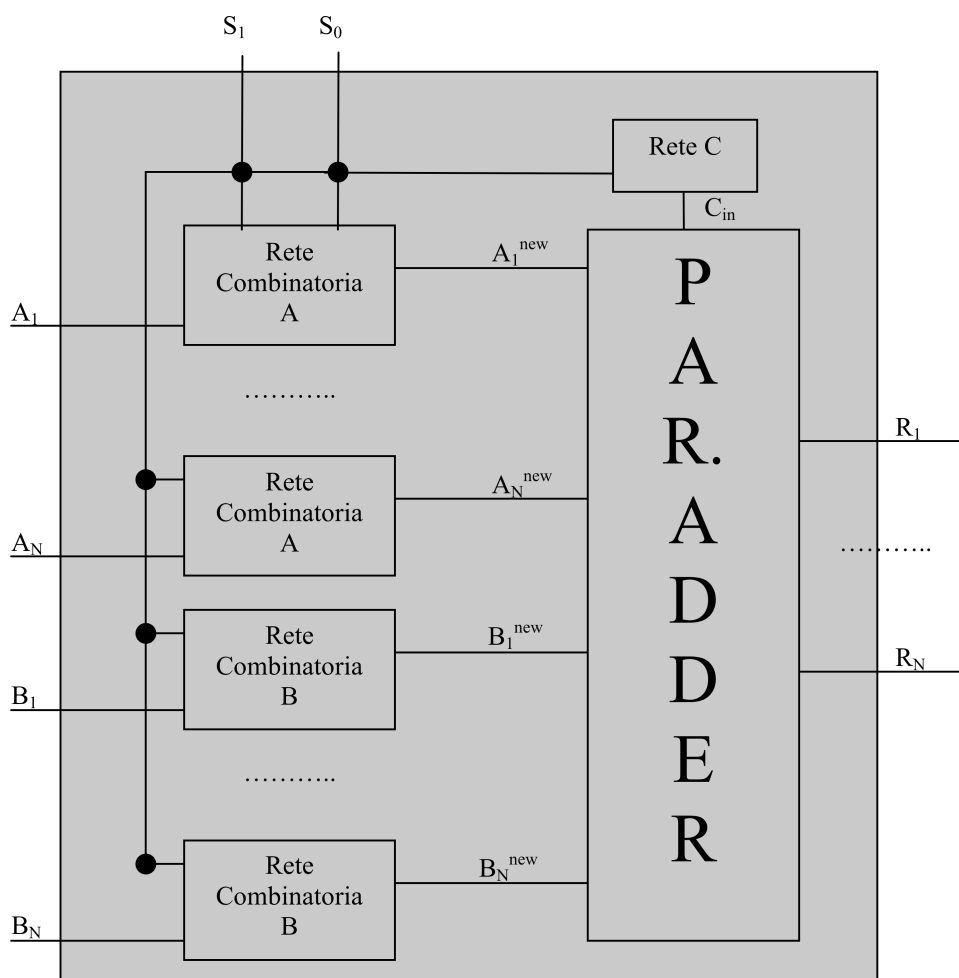
## ESERCIZIO 1

### Soluzione

1.

Lo schema progettuale è il seguente, in cui si devono progettare le reti logiche per trasformare i singoli bit di A, B, e calcolare il Cin da mandare in ingresso al parallel adder.

Gli ingressi delle reti per i singoli bit di A e B sono ovviamente  $A_i$  (o  $B_i$ ),  $S_0$  e  $S_1$ , mentre per la rete di Cin in ingresso avremo solo  $S_0$  e  $S_1$ .



Per quanto riguarda la specifica di progetto:

S0	S1	F
0	0	A+B
0	1	-A
1	0	A-1
1	1	A-B

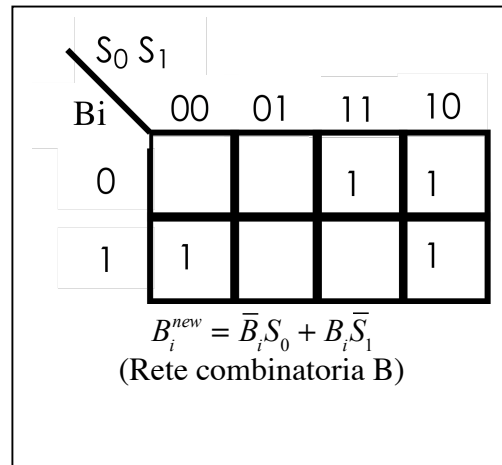
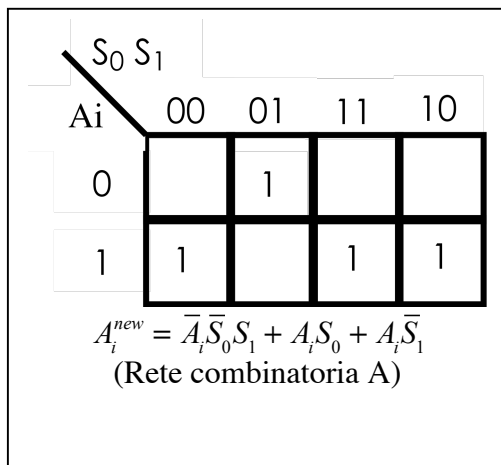
Per ottenere la funzione  $A+B$ , si dovrà fare in modo che  $A_{new} = A$ ,  $B_{new} = B$ ,  $C_{in} = 0$

Per ottenere  $-A$ , si dovrà fare in modo che  $A_{new} = A'$ ,  $B_{new} = 0$ ,  $C_{in} = 1$  (\*)

Per ottenere  $A-1$ , si dovrà fare in modo che  $A_{new} = A$ ,  $B_{new} = -1$ ,  $C_{in} = 0$

Per ottenere  $A-B$ , si dovrà fare in modo che  $A_{new} = A$ ,  $B_{new} = B'$ ,  $C_{in} = 1$  (\*)

(\*) l'apice indica il complemento a 1.



Infine, si nota che Cin = S1 (quindi non serve la rete combinatoria, è sufficiente collegare direttamente S1 al parallel adder).

2.

Si possono sostituire le reti combinatorie di A e B con dei mux 4-1 e un carry look ahead adder in luogo del parallel adder, in modo da minimizzare i tempi di propagazione dei riporti (relativamente ai full adder interni al parallel adder).

3.

Basta ricordare che nella rappresentazione di interi senza segno, il valore 111...1 su N bit vale  $2^N - 1$  e non -1.

Ne deriva che le operazioni eseguite saranno

S0	S1	F	F
0	0	A+B	A+B
0	1	-A	$2^N - A$
1	0	A-1	$A + 2^N - 1$
1	1	A-B	$A + 2^N - B$

## ESERCIZIO 2

### Soluzione

1. In modalità "DMA block transfer" il modulo DMA diventa il master del bus quindi tutti i trasferimenti da parte della CPU vengono sospesi. Si ha:  
max velocità di trasferimento =  $32 \text{ (bit/parola)} * (5 * 10^5) \text{ (parole/s)} = 2 * 10^6 \text{ B/s} = 1953,125 \text{ KB/s}$ .
2. In modalità "DMA transparent" il modulo DMA può intervenire solo quando alla CPU non occorre il bus di sistema. Quindi:  
max velocità di trasferimento =  $32 \text{ (bit/parola)} * [(2 * 0.8 + 5 * 0.2) * 10^5] \text{ (parole/s)} = 32 * 2.6 * 10^5 = 10.4 * 10^5 \text{ B/s} = 1015.625 \text{ KB/s}$ .
3. In modalità DMA con furto di ciclo, la frequenza dei furti equivale a  $2/7 * \text{freqCPU}$  (visto che vengono rubati 2 cicli ogni istruzione e che ogni istruzione richiede 5 cicli di clock, 2 cicli su 7 saranno utilizzati dal DMA per effettuare il trasferimento), per cui:  
max velocità di trasferimento =  $32 \text{ (bit/parola)} * 2/7 * 5 * 10^5 \text{ parole/s} = 558,07 \text{ KB/s}$ .
4. Si vedano le dispense del corso su memory mapped I/O e isolated I/O.

### ESERCIZIO 3

#### Soluzione

scalar\_product:

```
        addi $29, $29, -20
        sw $4, 0($29)
        sw $5, 4($29)
        sw $8, 8($29)
        sw $9, 12($29)
        sw $10,16($29)

        move $7, $0
        move $8, $0
for:     beq $8, $6, exit_for
        lw $9, 0($4)
        lw $10, 0($5)
        mul $9, $9,$10
        add $7, $7, $9
        addi $4, $4, 4
        addi $5, $5, 4
        addi $8, $8, 1
        j for

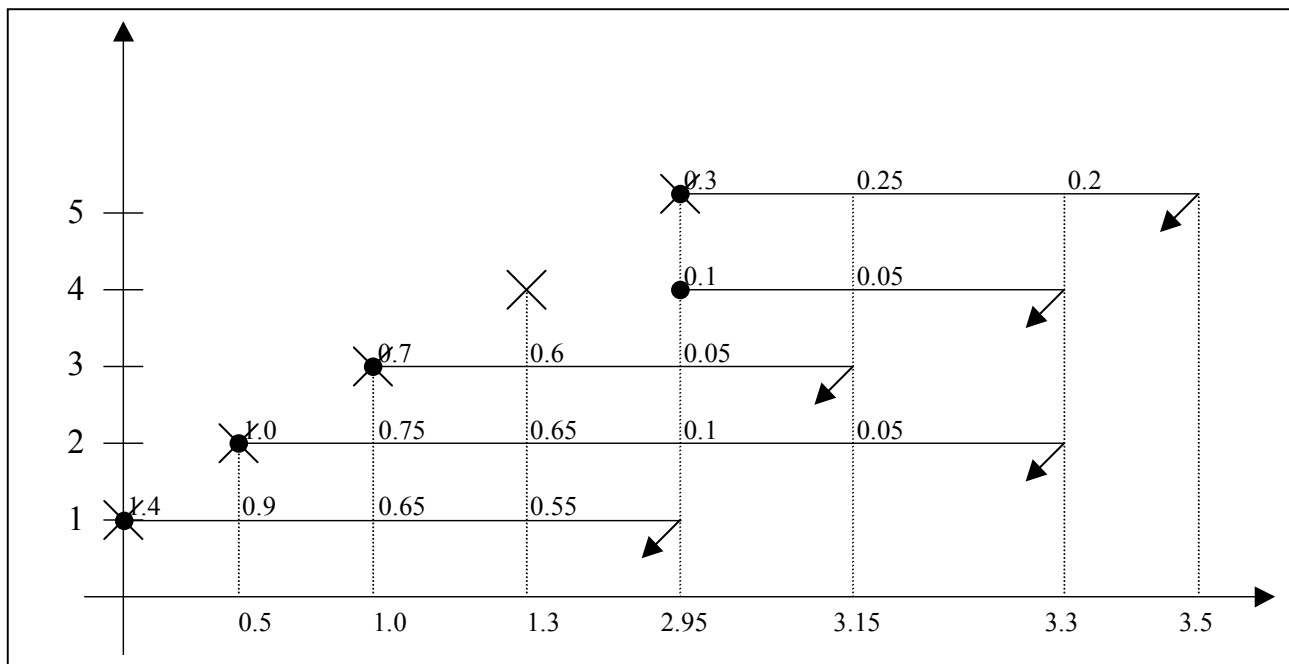
        #scalar_prod = 0
        #i=0
        #i==N, exit
        #$9 <- v[i]
        #$10 <- w[i]
        #$9 <- v[i] * w[i]
        #scalar_prod += v[i] * w[i]
        #incremento indirizzo v
        #incremento indirizzo w
        #i++

exit_for:
        lw $4, 0($29)
        lw $5, 4($29)
        lw $8, 8($29)
        lw $9, 12($29)
        lw $10,16($29)
        addi $29, $29, 20
        jr $31
```

## ESERCIZIO 4

### Soluzione.

1.



2.

#### Stato della memoria.

Istante 0.0: Crea partizione da 50K per il processo 1. Avvio del processo 1.

Istante 0.5: Crea partizione da 30K per il processo 2. Avvio del processo 2.

Istante 1.0: Crea partizione da 20K per il processo 3. Avvio del processo 3.

Istante 1.3: Non c'è spazio in memoria per allocare il processo 4, che viene posto in stato di attesa.

Istante 2.95: Il processo 1 ha termine, si alloca una partizione da 10K per il processo 4 che viene avviato. Arriva il processo 5 per cui si alloca una partizione da 20K. Avvio del processo 5.

Da questo istante i processi vengono gestiti regolarmente con FIFO multiprogrammata "round robin" fino al termine di tutti i processi.

Processo	Arrivo	Avvio	Termine	Turnaround	W.Tourn.
1	0.00	0.00	2.95	2.95	2.11
2	0.50	0.50	3.30	2.80	2.80
3	1.00	1.00	3.15	2.15	3.07
4	1.30	2.95	3.30	2.00	20.00
5	2.95	2.95	3.50	0.55	1.83
Media				2.09	5.96

3.

Si vedano i lucidi del corso.