CALCOLATORI ELETTRONICI A – 25 marzo 2009

NOME:	**COGNOME:**	**MATR:**
Scrivere chiaramente in caratteri maiuscoli a stampa		

1. Si elenchino le caratteristiche delle architetture (ISA):
   - ad accumulatore
   - a registri specializzati
   - load-store [3]

2. Scrivere una procedura in assembler MIPS corrispondente alla seguente funzione espressa in linguaggio C (F rappresenta una funzione chiamata dalla procedura che riceve un intero e produce un intero come risultato). Si utilizzino le note convenzioni sui registri. [5]

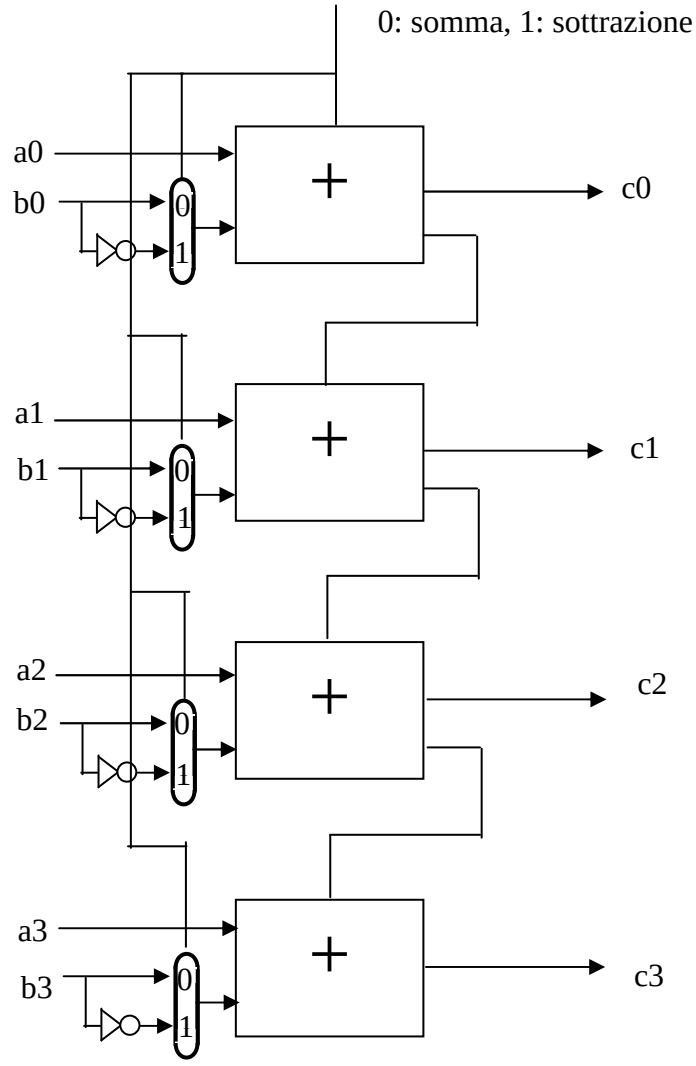
```
int P(int a){
        int temp;
        if(a>100) return a;
          else{
                temp=a;
                a=F(a);
                return temp+a;
                }
       }
```

Soluzione (ponendo temp nel registro s0)

```
P:     addi $sp, $sp, -8
       sw $s0, 4($sp)          // preserva (come chiamato) $s0 e $ra
       sw $ra, 0($sp)
       addi $t0, $zero, 100
       slti $t0, $t0, $a0      // if(a>100)…
       beq $t0, $zero, else
       add $vo, $a0, $zero     // valore di ritorno = a
       j fine
else:  add $s0, $a0, $zero     // temp=a
       jal F
       add $a0, $v0, $zero     // a=F(a)
       add $v0, $s0, $a0       // valore di ritorno = temp+a
fine:  lw $s0, 4($sp)
       lw $ra, 0($sp)
       addi $sp, $sp, 8
       jr $ra
```

3. Supponendo di avere a disposizione dei sommatori “full adder” (3 ingressi e due uscite) si riporti lo schema che permette di effettuare somme e sottrazioni a 4 bit. [3]

Soluzione



4. Si consideri la nota implementazione dell’unità di controllo secondo la tecnica multiciclo relativamente alle istruzioni MIPS lw, sw, beq, j e TIPO-R (si noti che il datapath è riportato nell’esercizio successivo). Si supponga che le operazioni atomiche che coinvolgono le unità funzionali principali richiedano:

Unità di memoria (lettura e scrittura): 2 ns
Register File (lettura e scrittura): 1 ns
Operazione ALU: 2 ns

Si ipotizzi che il carico di lavoro preveda in ogni caso, per le istruzioni sw e TIPO-R, una percentuale complessiva del 60%:

$f_{SW} + f_{TIPO\text{-}R} = 60\%$

Si chiede, riportando i passi significativi dell’analisi, di:

a) Determinare le condizioni in cui un’implementazione a singolo ciclo risulta più conveniente, in termini di prestazioni, rispetto all’implementazione multiciclo.

b) Rispondere alla stessa domanda supponendo che la lettura e la scrittura del Register File richiedano 2 ns al posto di 1 ns. [4]

Soluzione

a)
$T_{singolo\_ciclo}$=8ns (corrisponde alla lw)
$T_{multi}=10f_{lw}+8(f_{TIPOR}+f_{sw})+6(f_i+f_j)$
Singolo ciclo risulta migliore se $T_{singolo\_ciclo} < T_{multi}$
da cui deriva [tenendo conto che $f_{TIPOR}+f_{sw} = 0.6$ e $f_i+f_j=0.4- f_{lw}$]
$f_{lw}>0.2$

Quindi il singolo ciclo è migliore se la lw è eseguita con frequenza maggiore del 20%

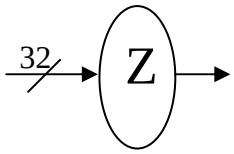
b) ripetendo i calcoli con i nuovi valori risulta $f_{lw}>70\%$, ma questo è impossibile dato che $f_{SW} + f_{TIPO-R} = 60\%$ (vedi testo dell'esercizio). Da ciò deriva che, nel secondo caso, il multiciclo è sempre migliore.
Ciò è evidente dal fatto che in questo caso tutte le fasi sono perfettamente bilanciate (durano 2ns) e quindi il singolo ciclo non può mai essere migliore del multiciclo.

5. Si considerino, mostrati nelle figure alla pagina seguente, il datapath ed il diagramma a stati finiti che specifica l'unità di controllo secondo la tecnica a multiciclo relativamente alle istruzioni MIPS *lw*, *sw*, *beq*, *j* ed alle istruzioni *Tipo-R*.
Si vuole implementare la nuova istruzione

MOVA r1, r2

che copia il vettore (array) di parole di memoria di indirizzo *r1* nell'indirizzo *r2*. Si suppone che la fine del vettore sia indicata da un elemento di valore nullo (che va anch'esso copiato).

Si supponga di avere a disposizione il seguente elemento funzionale combinatorio (con un tempo di propagazione trascurabile):

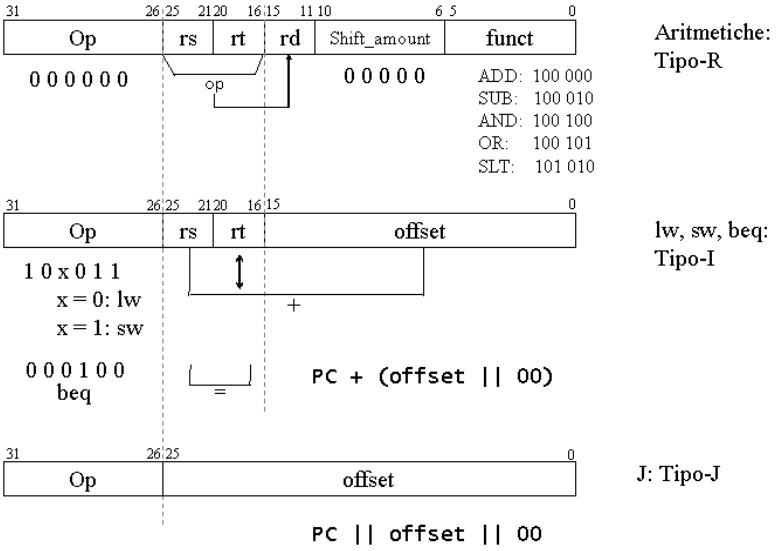


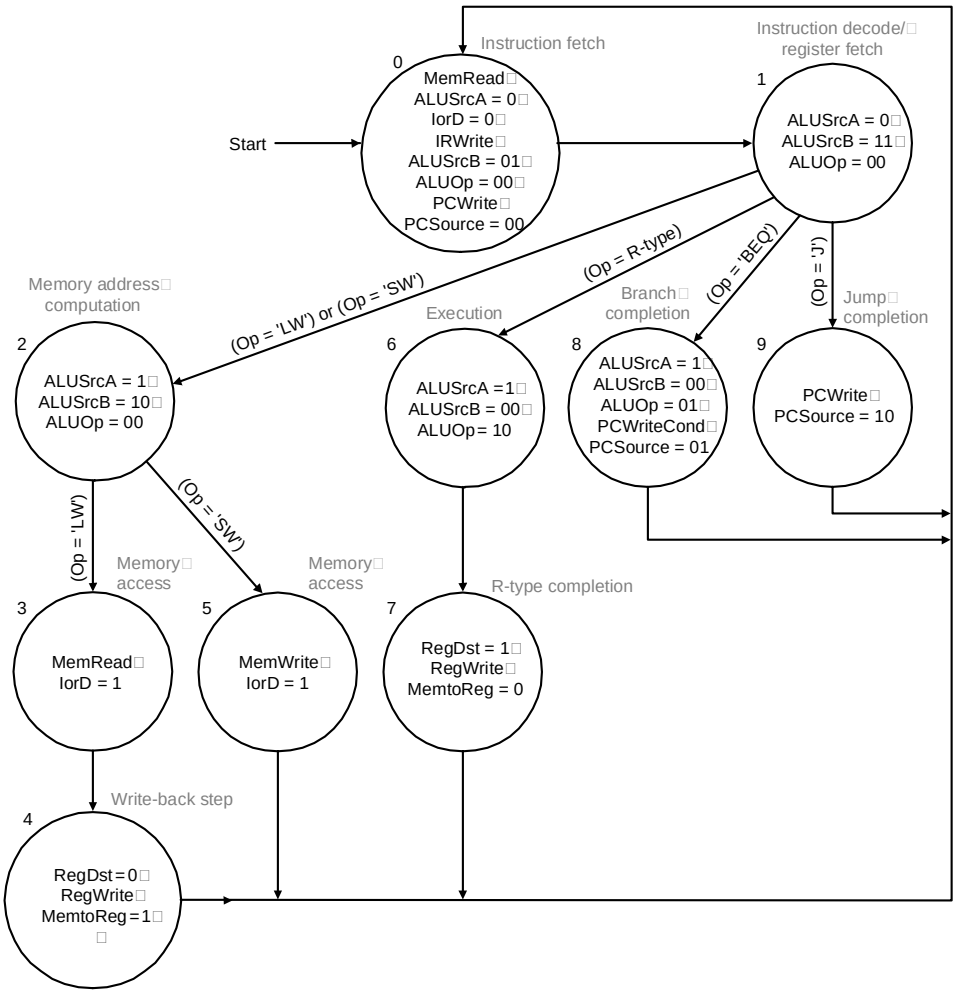
che riceve in ingresso un valore a 32 bit e restituisce 1 se tale elemento è nullo, 0 altrimenti.

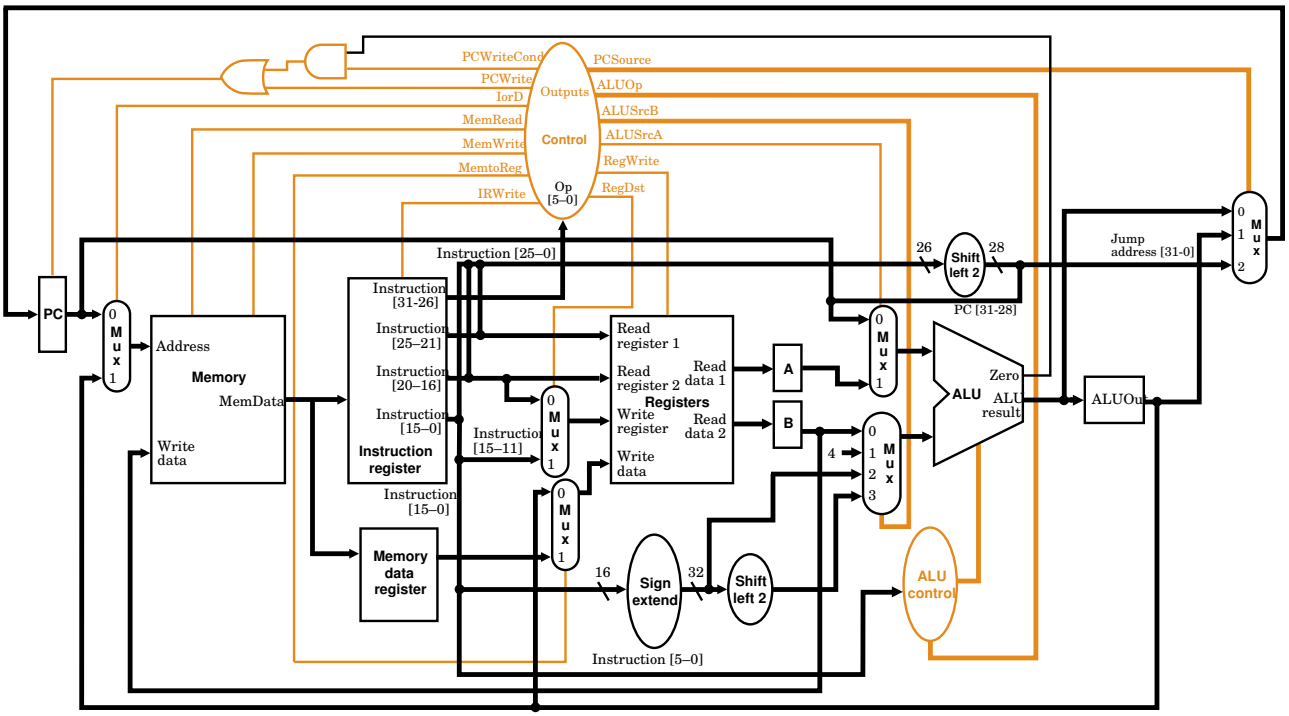
Ricordando i tre formati di codifica delle istruzioni (riportati di seguito) si chiede di:
- riportare il formato della nuova istruzione macchina (specificando anche i campi destinati rispettivamente a *r1* e *r2*);
- riportare, nella corrispondente figura, le modifiche necessarie al datapath;
- estendere il diagramma degli stati per implementare la nuova istruzione.

Infine, supponendo che il tempo di propagazione dell'elemento combinatorio non sia trascurabile ma diventi paragonabile a quello della ALU, indicare schematicamente come cambierebbe la soluzione individuata. [8]

<u>Promemoria formati delle istruzioni:</u>




Start
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
Instruction fetch
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
Instruction decode/ register fetch
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
Memory address computation
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10
Execution
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
Branch completion
9
PCWrite
PCSource = 10
Jump completion
(Op = 'LW')
(Op = 'SW')
3
MemRead
IorD = 1
Memory access
5
MemWrite
IorD = 1
Memory access
7
RegDst = 1
RegWrite
MemtoReg = 0
R-type completion
4
RegDst = 0
RegWrite
MemtoReg = 1
Write-back step


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op [5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
PC
Address
Memory
MemData
Write data
Instruction [31-26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–0]
Instruction register
Instruction [15–11]
Instruction [15–0]
Instruction [25–0]
Memory data register
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
A
B
4
ALU
Zero
ALU result
ALUOut
26
Shift left 2
28
PC [31-28]
Jump address [31-0]
16
Sign extend
32
Shift left 2
ALU control
Instruction [5–0]

Soluzione Si usa il formato I. Si tratta di caricare in A e B rs e rt rispettivamente. A e B conterranno gli indirizzi di memoria che andranno di volta in volta incrementati di 4.
Bisogna modificare il datapath e il diagramma degli stati in modo da creare un "ciclo" costituito da due stati:

S1: nel primo stato MDR=M[A] e A=A+4

S2: nel secondo stato M[B]=MDR e B=B+4; da questo stato l'istruzione termina se MDR=0, altrimenti si ritorna nel primo stato

Si veda l'allegato per la soluzione dettagliata.

Se il tempo di propagazione non è trascurabile, occorre aggiungere un registro a valle dell'elemento combinatorio Z e uno stato dopo S2, per evitare di mettere in serie tale elemento con l'unità di controllo.

6. Illustrare la differenza tra linking statico e linking dinamico. Quali vantaggi comporta il linking dinamico? [2]

7. Illustrare la politica di scrittura write-back per le memorie cache. Descrivere la funzione del dirty bit. Descrivere la funzione del write buffer e indicare precisamente quali parametri della cache contribuisce a migliorare e in quali situazioni entra in gioco. [5]

8. Si consideri una cache set-associativa a 4 vie in grado di memorizzare 64 KB (soltanto per la parte dati), cui si accede con indirizzi a 32 bit. Sapendo che i blocchi della cache contengono ciascuno 16 parole di 4 byte, determinare la dimensione totale in bit della memoria. [2]

Soluzione

Numero blocchi = 64kB/16*4 = 2^{10}
Numero insiemi = numero blocchi/4 = 2^{8}

L'offset è di 6 bit (ogni blocco ha 2^{6} byte), l'indice del set è di 8 bit, quindi l'etichetta è di 32-8-6=18 bit.

Dimensione della cache: 18*2^{10} (parte per le etichette) + 64kB = 530 kbit.

NB: non ho considerato il bit di validità, andava bene anche considerarlo e in tal caso la dimensione della cache sarebbe risultata
*19*2^{10} (parte per le etichette e bit di validità) + 64kB = 531 kbit*