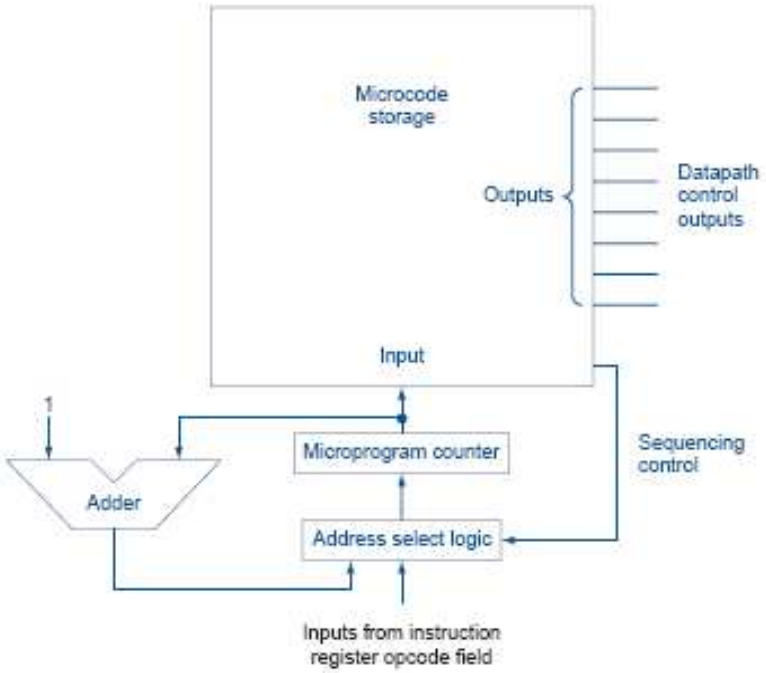
# CALCOLATORI ELETTRONICI B – 25 luglio 2007

| **NOME:** | **COGNOME:** |
|---|---|
| | **MATR:** |

Scrivere chiaramente in caratteri maiuscoli a stampa

1. Si consideri il seguente schema, che si riferisce ad una unità di controllo del processore microprogrammata. Per ciascuna delle seguenti affermazioni, si dica se è vera o falsa, motivando le risposte:
   - il contatore di microprogramma (microprogram counter) viene aggiornato quando viene aggiornato il program counter;
   - il contatore di microprogramma viene aggiornato più volte rispetto al program counter;
   - il contatore di microprogramma viene aggiornato ad ogni ciclo di clock. [2]



2. Si consideri il seguente frammento di codice MIPS:

```
sw      $s0, 20($t1)

add     $s0, $s0, $s0

lw      $s0, 40($s0)

sw      $s0, 40($t1)

add     $t1, $s0, $t1
```

Si consideri l'implementazione con pipeline a 5 stadi (F: Fetch, D: Decode, E: Execute, M: Mem, W: Write-Back). Si chiede di:
a) individuare in modo preciso tutte le dipendenze tra i dati
b) tracciare il diagramma temporale delle istruzioni (indicando esplicitamente le eventuali propagazioni e, per ognuna di esse, quale dato è propagato) in ognuna delle seguenti ipotesi:
- non è disponibile alcuna unità di propagazione
- è disponibile un'unità di propagazione verso lo stadio E
- è disponibile un'unità di propagazione verso lo stadio E ed una verso lo stadio M.

Nei diagrammi, si chiede di indicare il numero di cicli di penalità. [6]

3. Si consideri il processore dell'esercizio precedente, dotato di pipeline a 5 stadi, che disponga di un'unità di propagazione verso lo stadio E (come nel secondo caso al punto b dell'esercizio precedente).

   Il processore utilizza una cache primaria distinta per i dati e le istruzioni, mentre non dispone di cache secondaria. La cache, che in caso di successo consente di accedere all'istruzione o al dato in un ciclo di clock, presenta le seguenti caratteristiche:
   - percentuale di successo (hit rate): 90% per le istruzioni, 80% per i dati
   - penalità di fallimento in scrittura: 5 cicli di clock
   - penalità di fallimento in lettura: 10 cicli di clock

   Si assuma un carico di lavoro che prevede la seguente distribuzione delle istruzioni MIPS:

   | | |
   |---|---|
   | lw: | 30 % |
   | sw: | 10 % |
   | Tipo-R: | 40 % |
   | salti: | 20 % |

   tale che:
   - il 50% delle istruzioni che seguono lw non ne utilizzano il risultato, il 20% sono istruzioni di TIPO-R che ne utilizzano il risultato, il 5% sono sw che utilizzano il risultato della lw per il calcolo dell'indirizzo, il 5% sono sw che utilizzano il risultato sia per il calcolo dell'indirizzo sia per immagazzinarlo in memoria, il rimanente 20% sono sw che utilizzano il risultato della lw solo per immagazzinarlo in memoria.
   - il 10% delle istruzioni che seguono istruzioni di Tipo-R ne utilizzano il risultato nello stadio E, il 10% nello stadio M ed il rimanente 80% non ne utilizzano il risultato.

   Trascurando le criticità sui salti ma tenendo conto dei miss di cache e delle criticità sui dati, si calcoli il CPI (numero medio di cicli di clock per istruzione) ottenuto. [4]

4. Si consideri il seguente frammento di codice MIPS:

```
add   $s2, $t0, $t0

add   $s1, $s1, $s1

lw    $s1, 20($t0)

add   $s1, $s1, $s2

sub   $s4, $s1, $s3

add   $s1, $s3, $s3
```

Si consideri una implementazione con pipeline a 5 stadi (F: Fetch, D: Decode, E: Execute, M: Mem, W: Write-Back) in cui, per particolari esigenze di progetto, è presente una unità di propagazione verso lo stadio E e verso lo stadio M, ma non è presente alcuna unità di gestione delle criticità (ovvero, l'hardware non effettua stalli per gestire la dipendenza tra i dati). Si chiede, se possibile, di riordinare il codice in modo da garantirne una esecuzione corretta. Motivare la risposta con la maggior precisione possibile. [3]

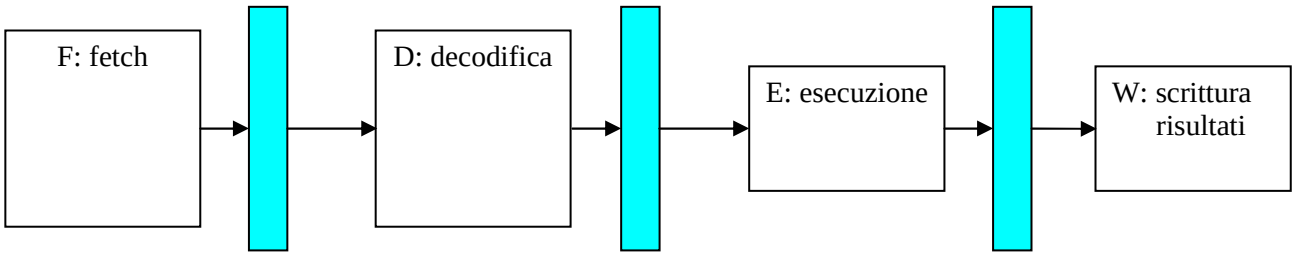
5. Si consideri il seguente frammento di codice MIPS (i puntini indicano la presenza di un certo numero di istruzioni non specificate):

```
lw    $t1, 20($t1)
beq   $t1, $t2, Dest
add   $s1, $t1, $t2
sub   $s2, $s1, $s2
…
Dest: …
```

Si consideri un'implementazione con pipeline a 5 stadi (Fetch – Decode – Execute – Mem – Write-Back) dotata di una coda delle istruzioni e con le seguenti caratteristiche:
- la coda può contenere 4 istruzioni, che l'unità di prelievo è in grado di caricare contemporaneamente in un solo ciclo di clock;
- i salti condizionati vengono eseguiti da un'apposita sotto-unità dell'unità di prelievo (Fetch), parallelamente all'esecuzione delle istruzioni negli altri stadi;
- per i salti condizionati, si utilizza la semplice tecnica di predizione statica di "salto non effettuato;
- l'hardware rende possibile la propagazione dei dati verso l'unità di prelievo delle istruzioni (e in particolare verso la sotto-unità che gestisce i salti condizionati).

Si chiede di:
- tracciare i diagrammi temporali nei due casi in cui la predizione per la beq sia corretta ed errata (assumere che le quattro istruzioni del frammento di codice siano caricate insieme nella coda delle istruzioni)
- determinare in entrambi i casi la penalità di salto. [4]

6. A partire dalla figura seguente, si disegnino schematicamente (ma in modo preciso) l'unità di controllo della pipeline ed i relativi collegamenti.
   Si consideri la presenza di una unità di propagazione verso E: da dove vengono prelevati i dati propagati verso E? [4]



7. E’ dato un bus sincrono che collega un processore P ad una memoria M e che consente il trasferimento di parole di memoria a blocchi. Il bus è dotato di linee condivise per dati e indirizzi (bus multiplexato). Le linee di controllo coinvolte in un’operazione di trasferimento di un blocco di parole dal processore alla memoria (scrittura) sono le seguenti:

   WRITE: utilizzato dal processore P per segnalare una richiesta di scrittura di un blocco. Esso rimane attivo fino a quando il processore desidera scrivere ulteriori parole (cfr. diagramma successivo).

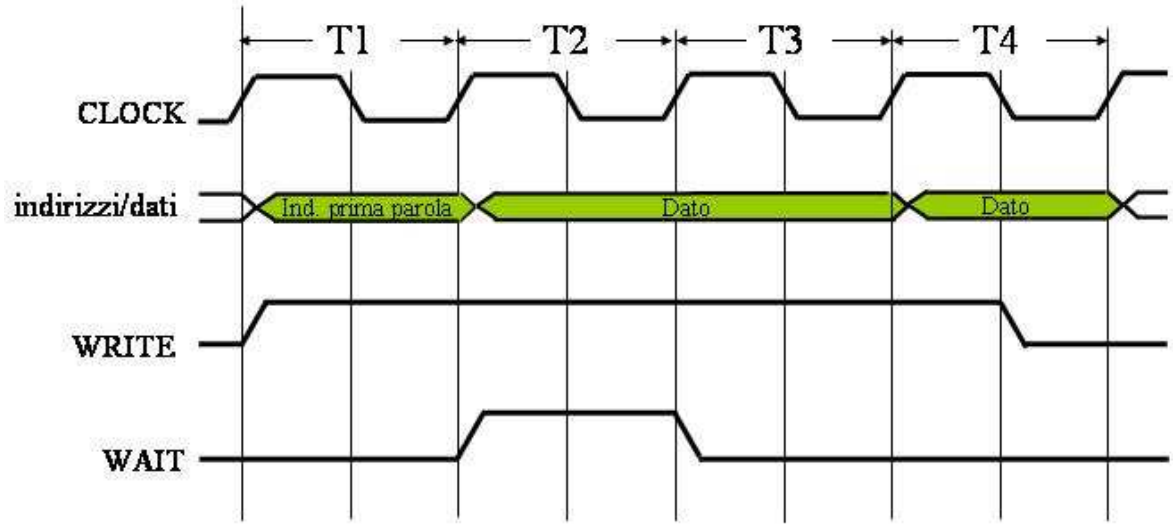
   WAIT: utilizzato dalla memoria per segnalare, prima dell’inizio del trasferimento, la necessità di uno o più cicli di clock di attesa (cfr. diagramma successivo).

   La figura seguente riporta l’evoluzione temporale di un’operazione di scrittura di un blocco costituito da due parole, supponendo che la memoria necessiti di un ciclo di clock di attesa.

   Si chiede di:

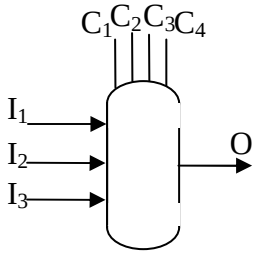
   a) Specificare la macchina a stati finiti che controlla l’esecuzione, nel processore P, del protocollo di scrittura di un blocco di due parole, assumendo che la memoria possa richiedere l’attesa di un numero arbitrario di cicli.
   b) Specificare la macchina a stati finiti che controlla l’esecuzione, nella memoria M, del protocollo di scrittura da parte del processore di un blocco di un numero generico (non nullo) di parole, assumendo che la memoria M necessiti di richiedere l’attesa di un ciclo di clock.

   Le specifiche proposte devono essere coerenti con il diagramma che segue. [6]



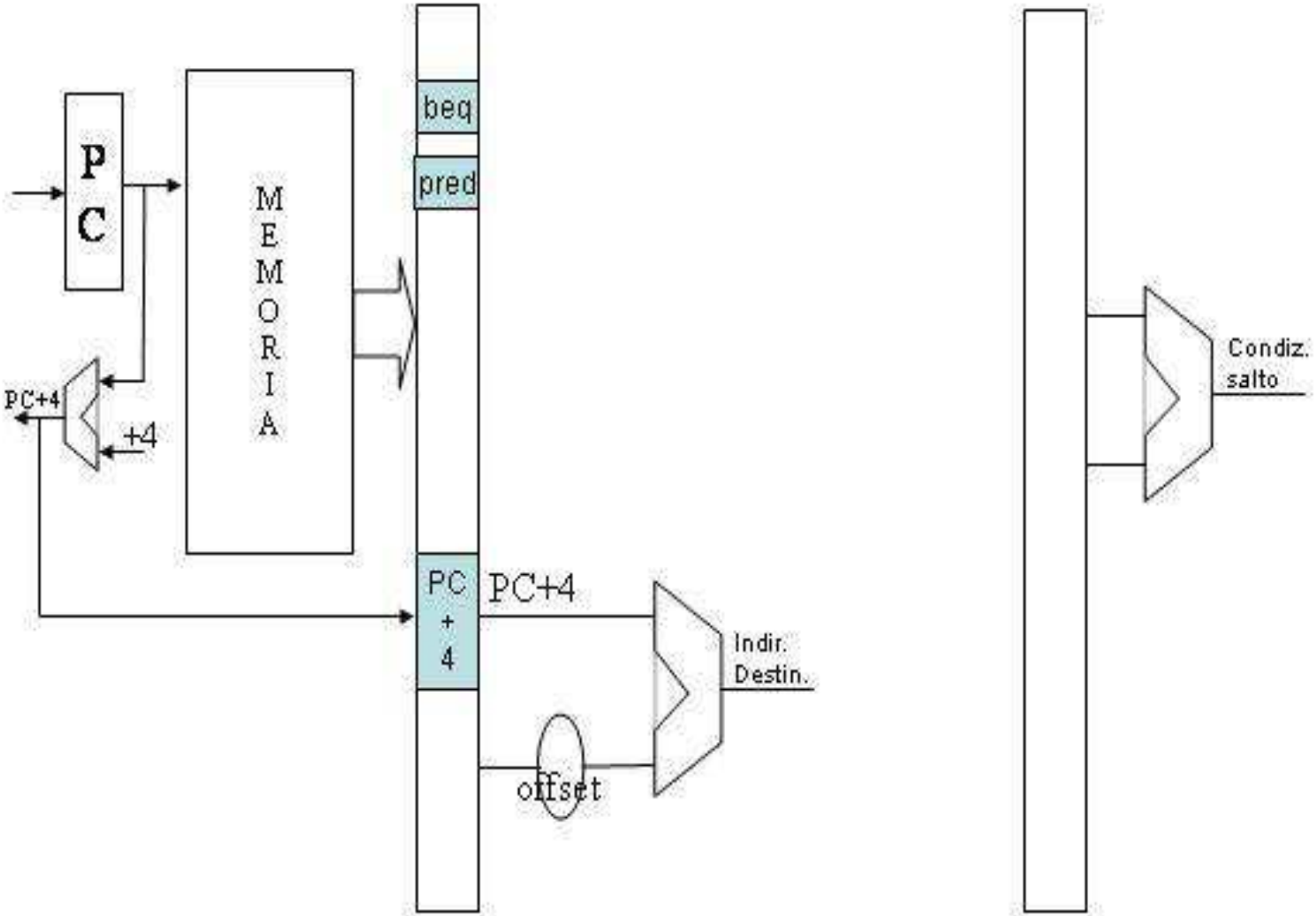
8. Si consideri l'implementazione di un processore con pipeline che utilizza la predizione statica dei salti condizionati basata sul compilatore. Si facciano le seguenti assunzioni:
   - nel formato macchina dell'istruzione sono previsti due specifici bit:
     *beq*: indica se l'istruzione è un salto condizionato (*beq* = 1: salto condizionato)
     *pred*: indica la predizione di salto (*pred* = 1: predizione di salto)
   - il calcolo della destinazione avviene nel secondo stadio D; nello stesso stadio viene effettuata la predizione (con l'eventuale aggiornamento del Program Counter)
   - la valutazione della condizione effettiva di salto viene effettuata nel terzo stadio E

   Si chiede di completare la figura nella pagina seguente (solo i primi tre stadi!) implementando la logica per la gestione dei salti condizionati. Per semplicità, è possibile utilizzare multiplexer *n-a-1* con *m* linee di controllo e logica di selezione dell'ingresso programmabile (da specificare in modo simile all'esempio seguente): [3]




$C_1 = C_2 = C_3 = C_4 = 0$: $O = I_1$
$C_1 = 1$: $O = I_1$
$C_2 = 1$: $O = I_2$
$C_3 = 1$ $C_4 = 0$: $O = I_3$

(si possono tralasciare le combinazioni don't care)


P
C
MEMORIA
beq
pred
PC+4
+4
PC
+
4
PC+4
Indir.
Destin.
offset
Condiz.
salto