Compito di Architettura dei Calcolatori - A.A. 2010-11
Prova di esame del 13 settembre 2011

Istruzioni: Spiegare chiaramente TUTTE le assunzioni che vengono effettuate per chiarire eventuali punti che si ritengono ambigui o non specificati.

1) [10 punti] Disegnare gli schemi di una periferica di I/O e del suo modulo di controllo, spiegare come sono connessi tra loro e con le altre parti del sistema, descrivere e spiegare quali flussi di dati si scambiano.

SVOLGIMENTO:

Si veda il cap. 7 del libro di testo e dei lucidi presentati a lezione, in particolare i lucidi da 3 a 7 (compresi).

2) [10 punti] Scrivere un programma nel linguaggio Assembly di Virtual CPU che realizza il seguente comportamento:

Si legge una locazione di memoria all'indirizzo A. Se in A si trova scritto 1 allora si esegue il blocco sotto specificato altrimenti si ritorna a controllare la locazione di memoria all'indirizzo A.

Il blocco da eseguire consiste nel contare quante volte in L>2 celle di memoria consecutive che iniziano all'indirizzo IND è vero che in due celle consecutive il valore contenuto nella prima è minore di o uguale al valore contenuto nella seconda. Se tale conteggio è maggiore o uguale a SOGLIA allora si scrive nella locazione di memoria di indirizzo OUT il valore -2 e ci si ferma. Altrimenti si torna al test principale su A. Si assume che in queste L celle di memoria i numeri sono rappresentati in notazione *non complementata*. Si noti che se ci sono 3 celle di memoria consecutive di indirizzo m1, m2, m3 i cui rispettivi valori v(m1), v(m2), v(m3) sono nella relazione $v(m1) \leq v(m2) \leq v(m3)$ allora il conteggio viene incrementato di 2.

I valori A, L, IND, OUT, SOGLIA sono contenuti nelle celle di memoria di indirizzo rispettivamente da 1 a 5. Commentare con adeguato dettaglio la logica seguita e le istruzioni usate

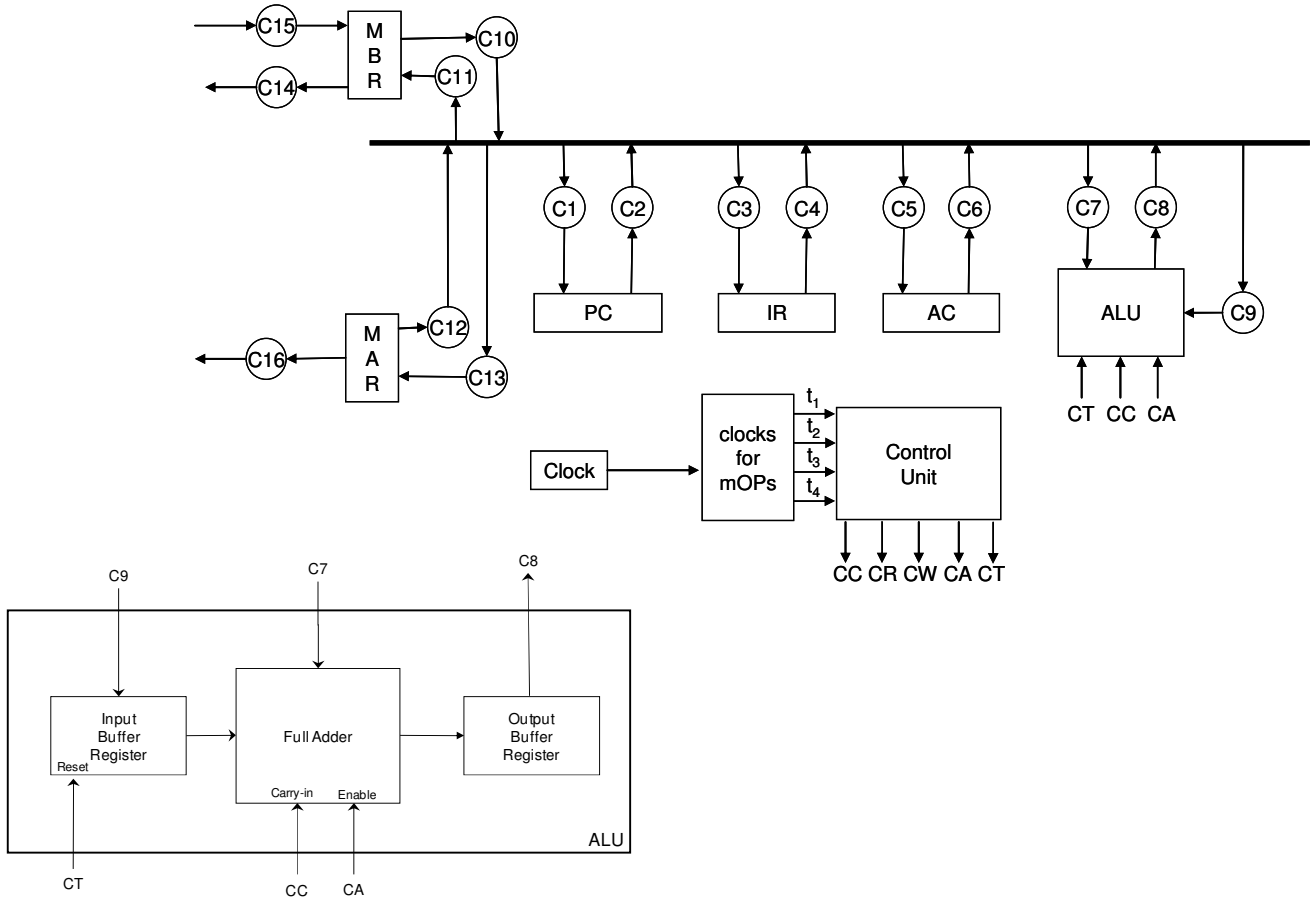
SVOLGIMENTO:

```
0    JMP 6         ; salta alla cella d'inizio del programma
; I DATI del programma, memorizzati in celle che sono usate come variabili di ingresso al programma
1                  ; contiene A, cioè l'indirizzo della locazione di memoria da verificare
2                  ; contiene L, cioè il numero di celle di memoria da verificare
3                  ; contiene IND, cioè l'indirizzo della prima cella da verificare
4                  ; contiene OUT, cioè l'indirizzo della cella di memoria in cui scrivere il risultato
5                  ; contiene SOGLIA, cioè il valore di controllo per il conteggio
; IL PROGRAMMA
; questo è il test iniziale
6    LOAD @@1      , si carica nell'accumulatore il valore della cella all'indirizzo A
7    DEC AX        ; gli si sottrae 1
8    JNZ 6         ; se all'indirizzo A c'è un valore diverso da 1 si riesegue il test
; inizializzazione del contatore CX che conteggia quante celle che soddisfano la relazione
9    LOAD 0
10   STORE CX
; inizializzazione del puntatore BX con l'indirizzo della cella corrente da esaminare
11   LOAD @3
12   STORE BX
; inizializzazione del contatore DX che conteggia quante coppie di celle consecutive vanno esaminate
13   LOAD @2
14   DEC AX
15   STORE DX
; ciclo che controlla tutte le coppie di celle.
16   LOAD @BX      ; si carica la cella corrente nell'accumulatore
17   INC BX
18   SUB @BX       ; si sottrae alla cella corrente (nell'accumulatore) la cella successiva
19   JG 21         ; se la cella corrente è maggiore della successiva la relazione non è verificata
20   INC CX        ; altrimenti si incrementa il conteggio di quante celle soddisfano la relazione
21   DEC DX        ; ho esaminato una coppia di celle quindi decremento DX
22   JNZ 16        ; se DX<>0 c'è ancora una coppia di celle da esaminare e torno all'inizio del ciclo
; qui ho esaminato tutte le coppie di celle e verifico se il conteggio è >= SOMMA
23   LOAD CX
24   SUB @5
25   JGE 27        ; se il conteggio è >= SOMMA vado a scrivere in OUT e terminare
26   JMP 6         ; altrimenti si ricomincia dal controllo sulla cella di indirizzo A
```

```
; qui ho finito, scrivo in OUT e termino
27   LOAD -2
28   STORE @@4
29   HLT
```

NB: Nel simulatore ENIAC l'istruzione JA non è implementata in modo corretto. Per questo motivo nella soluzione su esposta (che può essere testata sul simulatore ENIAC) alla riga 19 è stato usato JG, che è l'equivalente di JA per valori in versione complementata. Ovviamente in sede di esame si poteva benissimo usare JA. Inoltre, nel simulatore ENIAC l'istruzione JAE non è implementata. Per questo motivo nella soluzione su esposta alla riga 25 è stato usato JGE, che è l'equivalente di JAE per valori in versione complementata. Ovviamente in sede di esame si poteva benissimo usare JAE.

3) [10 punti] Dato lo schema della semplicissima CPU (VS0) sotto disegnato nella versione a singolo bus (con lo schema della relativa ALU) disegnare un nuovo schema che utilizza una struttura interna della CPU con due bus ed in cui sia la ALU che ogni registro hanno porte di lettura e scrittura su entrambi i bus. Descrivere inoltre, con riferimento a tale nuovo schema, le micro-operazioni ed il loro raggruppamento, sia per la fase di "fetch" che per la fase di "execute" delle quattro operazioni del repertorio di VS0 (cioè LOAD, STORE, ADD, JUMP).



SVOLGIMENTO:

Si rivedano, nell'appendice A dei lucidi presentati a lezione – in particolare i lucidi nelle pagine da 39 a 42 comprese, le micro-operazioni ed il loro raggruppamento per lo schema della semplicissima CPU "VS0" con struttura interna della CPU su singolo bus.

Il nuovo schema della CPU, con evidenziate le parti aggiunte è mostrato più sotto.

Le micro-operazioni ed il loro raggruppamento, sia per la fase di "fetch" che per quella di "execute" sono le seguenti:

Fetch:

t1:	MAR ← PC	C2	C13				
	PC+1	C17	C24	CT	CA	CC	(*questa micro-operazione poteva rimanere in t2 continuando ad usare il primo bus*)
t2:	MBR ← memory	C16	C15	CR			
	PC ← ALU	C8	C1				(*e in tal caso questa doveva rimanere in t3*)
t3:	IR ← MBR	C30	C20				

Nonostante la possibilità di anticipare temporalmente due micro-operazioni non si riesce ad ottenere per questa operazione una riduzione del tempo totale di esecuzione.

Execute LOAD:

t1:	MAR ← IR_{addr}	C4	C13		
t2:	MBR ← memory	C16	C15	CR	
t3:	AC ← MBR	C10	C5		

Per questa operazione la disponibilità del secondo bus è irrilevante. Quindi non si ottiene una riduzione del tempo totale di esecuzione.

Execute STORE:

t1:	MAR ← IR_{addr}	C4	C13		
	MBR ← AC	C21	C31		
t2:	memory ← MBR	C14	C16	CW	

Per questa operazione si ottiene una riduzione del tempo totale di esecuzione: viene completata in due unità di tempo invece che in tre come accade nello schema a bus singolo.

Execute ADD:

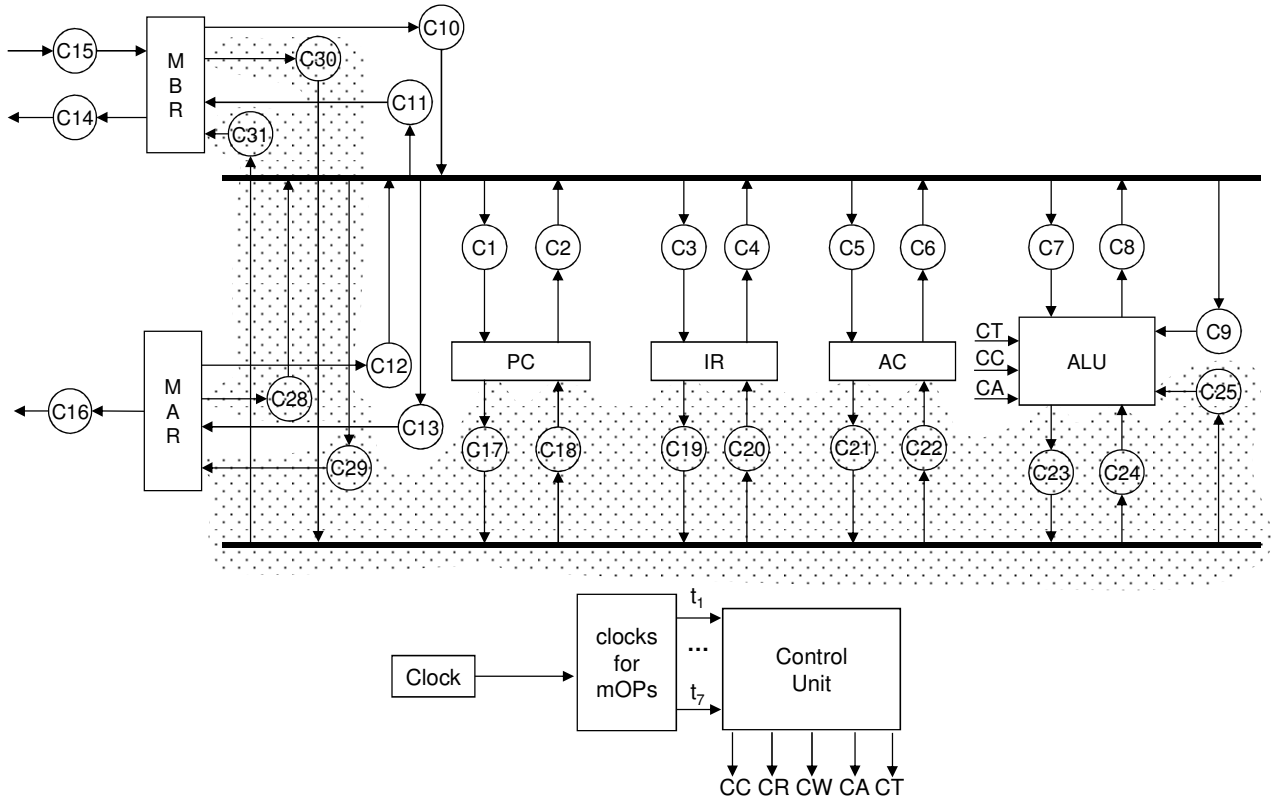
t1:	MAR ← IR_{addr}	C4	C13		
	ALU ← AC	C21	C25		(*questa micro-operazione poteva rimanere in t2 continuando ad usare il primo bus*)
t2:	MBR ← memory	C16	C15	CR	
t3:	MBR+ALU	C10	C7	CA	
t4:	AC ← ALU	C8	C5		

Nonostante la possibilità di anticipare temporalmente una micro-operazione non si riesce ad ottenere per questa operazione una riduzione del tempo totale di esecuzione.

Execute JUMP

t1:	PC ← IR_{addr}	C14	C2

Per questa operazione la disponibilità del secondo bus è irrilevante. Quindi non si ottiene una riduzione del tempo totale di esecuzione.


C15
C14
MBR
C30
C31
C10
C11
C1
C2
C3
C4
C5
C6
C7
C8
C9
CT
CC
CA
ALU
C25
PC
IR
AC
C12
C16
MAR
C28
C13
C29
C17
C18
C19
C20
C21
C22
C23
C24

Clock
clocks for mOPs
t1
...
t7
Control Unit
CC CR CW CA CT