Domande

1. (punti: -1, 4)
   Una microarchitettura di tipo microprogrammata:

   (a) presenta normalmente un "data path" più complesso rispetto a quello di una equivalente microarchitettura RISC;

   (b) richiede l'uso di una Control Store per contenere il microcodice che, interpretato, permette la realizzazione delle istruzioni della macchina convenzionale;

   (c) richiede l'uso di un registro delle microistruzioni (MIR) nel quale memorizzare il codice operativo della prossima istruzione della macchina convenzionale da interpretare;

   (d) viene detta ad organizzazione orizzontale quando la Control Store assume dimensioni rilevanti a causa dell'elevato numero di bit delle parole di memoria (architetture a 32 o 64 bit);

   (e) viene detta ad organizzazione verticale quando la dimensione della Control Store viene minimizzata riducendo il numero di istruzioni della macchina convenzionale;

   (f) nessuna delle affermazioni precedenti é corretta.

2. (punti: -1, 4)
   La memoria virtuale a paginazione e segmentazione:

   (a) é un meccanismo di traduzione di indirizzi fisici in indirizzi virtuali realizzato esclusivamente a livello hardware dal dispositivo MMU;

   (b) comporta l'uso di dispositivi hardware per la traduzione efficiente da indirizzi virtuali ad indirizzi fisici in RAM;

   (c) necessita una fase di gestione della memoria da parte del sistema operativo per mantenere la corrispondenza tra indirizzi virtuali ed indirizzi fisici durante l'esecuzione di un programma;

   (d) permette di sostituire l'accesso a dati in memoria RAM con l'accesso a dati contenuti sulle memorie di massa (dischi di grandi dimensioni) in modo trasparente rispetto all'esecuzione di un programma da parte della CPU;

   (e) richiede una scelta a livello architetturale sulla dimensione delle pagine e dei segmenti, senza poi possibilità di variazione in fase di operazione del sistema;

   (f) nessuna delle affermazioni precedenti é corretta.

3. (punti: -1, 4)
   Un codice ridondante a rilevazione e correzione di errori:

   (a) é basato sul concetto di distanza di Hamming per suddividere le codifiche corrette da quelle erronee e, in caso di errore, individuare la codifica corretta più vicina a quella errata considerata;

   (b) richiede l'uso di almeno $n + 1$ bit di parità per riconoscere la presenza di n errori;

   (c) richiede l'uso di almeno $2n + 1$ bit di parità per correggere una codifica contenente n errori;

   (d) se applicato alla memoria RAM consente di ovviare alla presenza di alcuni semplici errori di programmazione che producono errori su un numero limitato di bit nella rappresentazione dei dati;

   (e) non é convenientemente realizzabile in hardware a causa della complessità dell'algoritmo di correzione;

   (f) nessuna delle affermazioni precedenti é corretta.

4. (punti: -1, 4)
Il modo di indirizzamento “immediato”:

(a) permette di accedere al contenuto di una cella di memoria mediante un indirizzo costante, specificato nel codice operativo dell’istruzione;

(b) determina un accesso in sola lettura al segmento dati, garantendo quindi una corretta lettura di valori costanti, impedendo fisicamente al processore di alterarli;

(c) nel caso di un processore con codifica di tutte le istruzioni usando una sola cella di memoria consente un tempo di accesso inferiore rispetto al modo “registro”, in quanto il dato costante risulta essere disponibile già al termine della decodifica dell’istruzione, senza aspettare la fase di esecuzione;

(d) trova un esempio di realizzazione nella istruzione POPI della macchina VM-2;

(e) trova un esempio di realizzazione nella istruzione LODL della macchina VM-2;

(f) nessuna delle affermazioni precedenti é corretta.

5. (punti: -1, 4)
Una memoria cache da 32KB organizzata in 4 insiemi associativi con parole da 32 bit, linee da 2 parole, indirizzamento a livello di byte fino ad un massimo di 256MB:

(a) richiede per la realizzazione del meccanismo di ricerca associativa sul campo “tag” l’uso di 4 comparatori da 15 bit e di una memoria RAM da 1024 celle da 60 bit;

(b) richiede per la realizzazione del meccanismo di ricerca associativa sul campo “tag” l’uso di 4 comparatori da 16 bit e di 4 memorie RAM da 1024 celle da 16 bit;

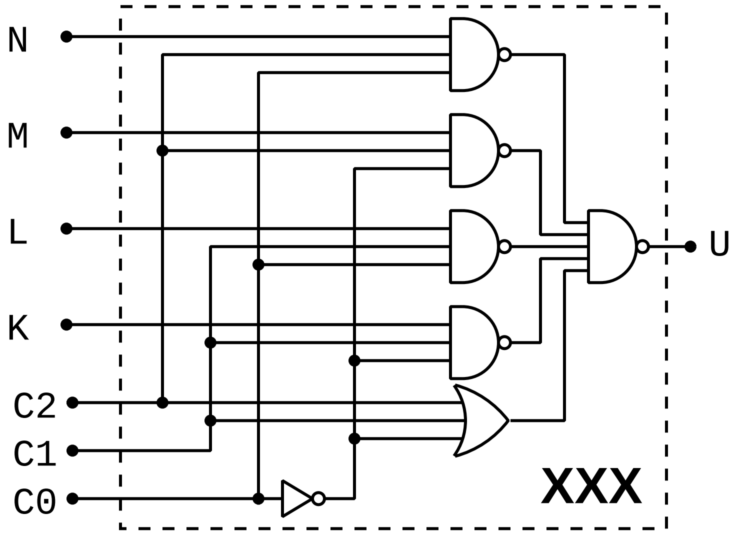
(c) consente di mappare qualsiasi byte della RAM in 2 celle diverse della cache;

(d) consente di mappare qualsiasi byte della RAM in 4 celle diverse della cache;

(e) consente di mappare qualsiasi byte della RAM in 8 celle diverse della cache;

(f) consente di mappare qualsiasi byte della RAM in 1024 celle diverse della cache;

6. Considerare il seguente modulo circuitale XXX con 7 ingressi booleani ($C0$, $C1$, $C2$, K, L, M e N) e 1 uscita (U):



(a) (punti: -1, 1)
Il circuito é di tipo Combinatorio, Sequenziale Asincrono o Sequenziale Sincrono?

(b) (punti: 4) ricavare i valori dell'uscita data la seguente sequenza di valori in ingresso:

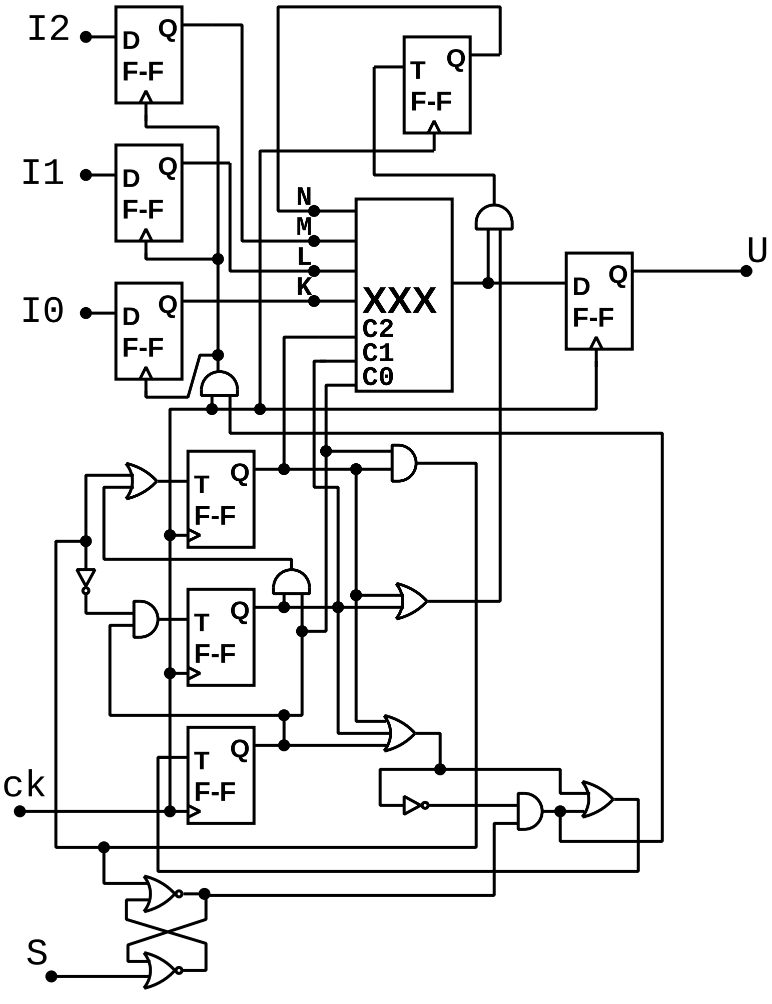
n.	$C2$	$C1$	$C0$	K	L	M	N	U
1	0	0	0	1	1	1	1	
2	0	0	1	0	0	0	0	
3	0	0	1	0	0	1	1	
4	0	1	0	0	0	1	1	
5	0	1	0	1	1	0	0	
6	0	1	1	1	0	0	0	
7	0	1	1	1	0	1	0	
8	0	1	1	0	0	1	1	

(c) (punti: 4)

Disegnare la rappresentazione BDD canonica della funzione U di uscita, assumendo l'ordinamento delle variabili così come elencato da sinistra verso destra nella tabella al punto (b).

(d) (punti: 13)

Considerare il seguente circuito sequenziale sincrono, contenente il modulo circuitale XXX precedentemente illustrato:



Supponendo che il circuito parta dallo stato in cui tutti i flip-flop assumono il valore 0, riempire la tabella che riporta i valori delle uscite in funzione dei primi 6 cicli di clock dopo aver presentato sugli ingressi $I2$, $I1$, $I0$ la codifica binaria della cifra meno significativa del vostro numero di matricola modulo 8 ed inviato un impulso 1 sull'ingresso S:

$I2 =$, $I1 =$, $I0 =$

$U(1) =$, $U(2) =$, $U(3) =$, $U(4) =$, $U(5) =$, $U(6) =$

7. (punti: 14)
Considerare la seguente procedura "ppp" definita in Assembler SPARC:

```
ppp:    save %sp,-112,%sp
        ldub [%i0],%l3
        ldub [%i0+6],%l2
        mov %i0,%l0
        cmp %l3,%l2
        bleu .LL2
        add %l0,6,%l1
        mov %l2,%o1
        stb %o1,[%l0]
        mov %l3,%l2
        stb %l2,[%l1]
        mov %o1,%l3
.LL2:   mov 0,%o1
        b .LL3
        mov 0,%l4
.LL5:   cmp %l5,%l3
        bgeu .LL6
        add %o1,1,%o1
        stb %l3,[%l0]
        add %l4,%l3,%l4
        stb %l5,[%i0]
        b .LL3
        mov %l5,%l3
.LL6:   cmp %l5,%l2
        bleu,a .LL3
        add %l4,%l5,%l4
        stb %l2,[%l0]
        add %l4,%l2,%l4
        stb %l5,[%l1]
        mov %l5,%l2
.LL3:   add %l0,1,%l0
        cmp %l0,%l1
        blu,a .LL5
        ldub [%l0],%l5
        call .udiv,0
        mov %l4,%o0
        mov %o0,%l4
        mov %i0,%l0
        cmp %l0,%l1
        bgu .LL12
        stb %l4,[%i1]
```

```
        ldub [%l0],%o0
.LL17:  cmp %o0,%l4
        bgeu .LL13
        add %l0,1,%l0
        call printunsigned,0
        nop
.LL13:  cmp %l0,%l1
        bleu,a .LL17
        ldub [%l0],%o0
.LL12:  ret
        restore
```

ppp viene chiamata da un programma principale il quale inserisce due parametri nei registri:
in %o0 l'indirizzo del primo elemento di un vettori di 7 byte, contenente la codifica binaria delle sette cifre del vostro numero di matricola (nell'ordine);
in %o1 l'indirizzo di un ulteriore byte allocato in memoria ed inizializzato al valore 0.
Scrivere la sequenza dei valori che verrebbero stampati dall'esecuzione della procedura coi vostri dati, seguito dal valore finale assunto dalla variabile il cui indirizzo é stato passato nel registro %o1.

8. (punti: 12)
Completare la seguente traduzione della procedura ppp della domanda precedente in assembler VM-2 con le 4 istruzioni mancanti alle etichette X1, X2, X3 e X4.

```
ppp:  LODL 2
      PUSH
      PSHI
 X1:
      ADDL 2
      PUSH
      PSHI
      LODL -3
      SIGN
      ADDL -5
      JPOS LL2
      LODL -5
      PUSH
      PUSH
      LODL -2
      POPI
      LODL -3
      STOL -5
      PUSH
      LODL -4
      POPI
      POP
      STOL -3
LL2:  LOC8 0
```

```
        PUSH
        PUSH
        JUMP LL3
LL5:    LODL -2
        PSHI
        LOC8 1
        ADDL -6
        STOL -6
        LODL -3
        SIGN
        ADDL  -8
 X2:
        LODL -3
        PUSH
        LODL -2
        POPI
        LODL -7
        ADDL -3
        STOL -7
 X3:
        STOL -3
        LODL 2
        POPI
        JUMP LL3
LL6:    LODL -8
        SIGN
        ADDL -5
        JNEG LL4
 X4:
        ADDL -7
        STOL -7
        JUMP LL3
LL4:    LODL -5
        PUSH
        LODL -2
        POPI
        LODL -7
        ADDL -5
        STOL -7
        LODL -8
        STOL -5
        LODL -4
        POPI
LL3:    LOC8 1
        ADDL -2
        STOL -2
        LODL -4
        SIGN
        ADDL -2
        JNEG LL5
```

```
        LODL -6
        PUSH
        CALL .udiv
        STOL -7
        STOL -8
        LODL 1
        POPI
        LODL 2
        STOL -2
        SIGN
        ADDL -4
        JNEG LL12
LL17:   LODL -2
        PSHI
        LOC8 1
        ADDL -2
        STOL -2
        LODL -7
        SIGN
        ADDL -8
        JPOS LL13
        CALL printunsigned
LL13:   LODL -2
        SIGN
        ADDL -4
        JPOS LL17
LL12:   RETN
```

I parametri in ingresso a `ppp` vengono in questo caso passati sullo stack, così come quelli alla funzione `.udiv` ed alla procedura `printunsigned`. Il valore calcolato da `.udiv` viene ritornato alla procedura `ppp` nel registro Accumulatore.

Risposte Scritto di Architettura, 25/9/1996

Cognome : **Nome:** **Matricola:**

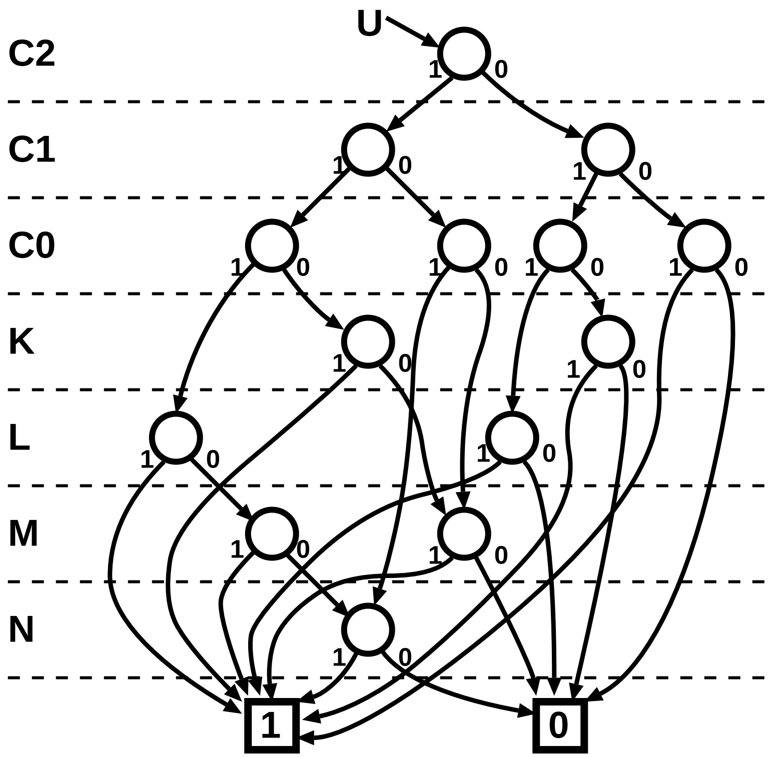
1. (punti: -1, 4) **a** **X b** **c** **d** **e** **f**
2. (punti: -1, 4) **a** **X b** **X c** **d** **e** **f**
3. (punti: -1, 4) **X a** **b** **c** **d** **e** **f**
4. (punti: -1, 4) **a** **b** **c** **d** **e** **X f**
5. (punti: -1, 4) **X a** **b** **c** **X d** **e** **f**
6.

(a) (punti: -1, 1) X Combinatorio Asincrono Sincrono

(b) (punti: 4)

n.	$C2$	$C1$	$C0$	K	L	M	N	U
1	0	0	0	1	1	1	1	0
2	0	0	1	0	0	0	0	1
3	0	0	1	0	0	1	1	1
4	0	1	0	0	0	1	1	0
5	0	1	0	1	1	0	0	1
6	0	1	1	1	0	0	0	0
7	0	1	1	1	0	1	0	0
8	0	1	1	0	0	1	1	0

(c) (punti: 4)



(d) (punti: 13)

$I2 =$, $I1 =$, $I0 =$

$U(1) = 0$, $U(2) = 1$, $U(3) = I0$, $U(4) = I1$, $U(5) = I2$, $U(6) = I0 \oplus I1 \oplus I2$

7. (punti: 14) Calcola il valor medio eliminando il valore più alto e quello più basso, poi stampa tutti i valori minori del valor medio determinato.
   Esempio: matricola= 1 3 5 9 6 4 2, valor medio = 4, valori stampati= 1 2 3

8. (punti: 12)

```
        ...
        PSHI
X1:     LOC8 6
        ADDL 2
        ...
        ADDL  -8
X2:     JPOS LL6
        LODL -3
        ...
        STOL -7
X3:     LODL -8
        STOL -3
        ...
        JNEG LL4
X4:     POP
        ADDL -7
        ...
```

Risposte Scritto di Architettura, 25/9/1996

Cognome : **Nome:** **Matricola:**

1. (punti: -1, 4) **a** **b** **c** **d** **e** **f**
2. (punti: -1, 4) **a** **b** **c** **d** **e** **f**
3. (punti: -1, 4) **a** **b** **c** **d** **e** **f**
4. (punti: -1, 4) **a** **b** **c** **d** **e** **f**
5. (punti: -1, 4) **a** **b** **c** **d** **e** **f**
6.

   (a) (punti: -1, 1) Combinatorio Asincrono Sincrono

   (b) (punti: 4)

n.	$C2$	$C1$	$C0$	K	L	M	N	U
1	0	0	0	1	1	1	1	
2	0	0	1	0	0	0	0	
3	0	0	1	0	0	1	1	
4	0	1	0	0	0	1	1	
5	0	1	0	1	1	0	0	
6	0	1	1	1	0	0	0	
7	0	1	1	1	0	1	0	
8	0	1	1	0	0	1	1	

   (c) (punti: 4)

   (d) (punti: 13)

   $I2 =$, $I1 =$, $I0 =$

   $U(1) =$, $U(2) =$, $U(3) =$, $U(4) =$, $U(5) =$, $U(6) =$

7. (punti: 14) valori=

8. (punti: 12)

```
        ...
        PSHI
X1:
        ADDL 2
        ...
        ADDL  -8
X2:
        LODL -3
        ...
        STOL -7
X3:
        STOL -3
        ...
        JNEG LL4
X4:
        ADDL -7
        ...
```