Domande

1. (punti: -1, 4)
   L'Unità di gestione della memoria (MMU):

   (a) fornisce alla CPU la traduzione di indirizzi logici in indirizzi fisici; normalmente i controller dei dispositivi di I/O lavorano con indirizzamento fisico quando accedono in DMA;

   (b) fornisce sia alla CPU sia a tutti i controller abilitati a diventare master del bus la traduzione di indirizzi logici in indirizzi fisici;

   (c) é realizzata mediante l'uso di una memoria completamente associativa che contiene le tabelle di traduzione da pagine logiche a pagine fisiche;

   (d) é realizzata mediante l'uso di una memoria cache a corrispondenza diretta che contiene le tabelle di traduzione da pagine logiche a pagine fisiche;

   (e) può essere usata solo in associazione con una memoria cache di secondo livello che contiene le tabelle di traduzione da pagine logiche a pagine fisiche;

   (f) nessuna delle affermazioni precedenti é corretta.

2. (punti: -1, 4)
   Il tipo di dato "tagged" supportato dalla architettura SPARC:

   (a) rappresenta numeri interi senza segno su 30 bit; in fase di esecuzione la CPU genera una condizione di errore nel caso il valore non sia del tipo previsto;

   (b) viene usato per la definizione di formati diversi di rappresentazione in memoria (per esempio su 8, 16 o 32 bit) per numeri interi; il meccanismo delle trap consente di gestire correttamente le eventuali operazioni di estensione del segno;

   (c) supporta in modo efficiente linguaggi nei quali il tipo dei dati é noto solo in fase di esecuzione, e nei quali il tipo più frequentemente usato é l'intero rappresentato su 30 bit;

   (d) permette di realizzare operazioni di somma e sottrazione su dati non interi mediante l'attivazione di una trap;

   (e) non consente l'uso di un codice ad espansione per la rappresentazione di interi su più di 30 bit;

   (f) nessuna delle affermazioni precedenti é corretta.

3. (punti: -1, 4)
   Il protocollo di "handshaking" a due variabili booleane:

   (a) viene comunemente impiegato per la realizzazione di bus di tipo sincrono;

   (b) richiede una connessione di tipo bidirezionale tra i due partners coinvolti nella comunicazione;

   (c) richiede a ciascun partner nella comunicazione la conoscenza dei tempi di risposta dell'altro per portare a termine correttamente la comunicazione;

   (d) rispetto ad altre tecniche richiede l'uso di due fili aggiuntivi sul bus per la sua realizzazione;

   (e) richiede l'individuazione di un "master" che prende l'iniziativa della comunicazione e di uno "slave" che può solo rispondere alle richieste del master;

   (f) nessuna delle affermazioni precedenti é corretta.

4. (punti: -1, 4)
   Il modo di indirizzamento "indiretto mediante registro":

   (a) permette di accedere al contenuto di un registro e di copiarne il contenuto in una cella di memoria di indirizzo specificato;

(b) permette di accedere al contenuto di una cella di memoria caricandone precedentemente l'indirizzo in un registro;

(c) permette di leggere (indirettamente) il contenuto di una cella di memoria accedendo invece ad un registro che ne mantiene una copia aggiornata;

(d) trova un esempio di realizzazione nella istruzione POPI della macchina VM-2 (facendo riferimento al registro Accumulatore);

(e) trova un esempio di realizzazione nella istruzione LODL della macchina VM-2 (facendo riferimento al registro Frame Pointer);

(f) nessuna delle affermazioni precedenti é corretta.

5. (punti: -1, 4)
La Tabella dei Simboli di un programma Assemblatore organizzato in due passi:

(a) viene completata nel secondo passo con le informazioni relative agli indirizzi delle etichette nel segmento codice, dopo che questo é stato generato nel primo passo;

(b) viene riempita nel primo passo con le parole chiave del linguaggio Assembler specifico per il processore considerato, e completata nel secondo passo con le etichette simboliche specificate dal programmatore;

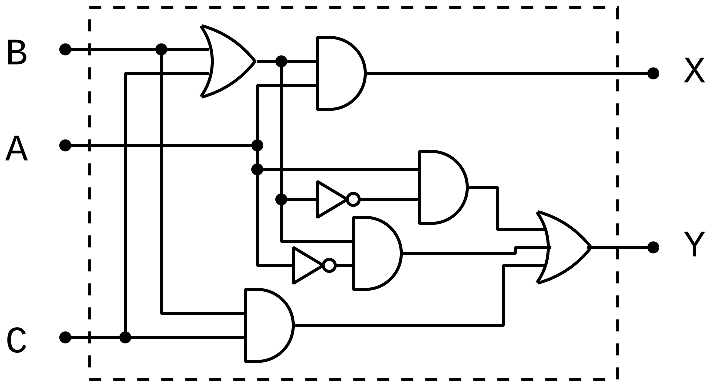
(c) viene completata nel primo passo con le informazioni relative alle etichette simboliche specificate dal programmatore per poter essere usata nel secondo passo di traduzione da simboli ad indirizzi (virtuali);

(d) viene solitamente suddivisa in simboli locali, importati ed esportati ad uso del programma "linker" per facilitarne le operazioni di rilocazione del codice;

(e) viene solitamente suddivisa in simboli locali, importati ed esportati ad uso del programma "loader" per facilitarne le operazioni di chiamata di funzioni di librerie dinamiche secondo una strategia di valutazione "lazy";

(f) nessuna delle affermazioni precedenti é corretta.

6. Considerare il seguente modulo circuitale con 3 ingressi booleani (A, B e C) e 2 uscite booleane (X e Y):



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

(b) (punti: 4) ricavare i valori delle uscite data la seguente sequenza di valori in ingresso:

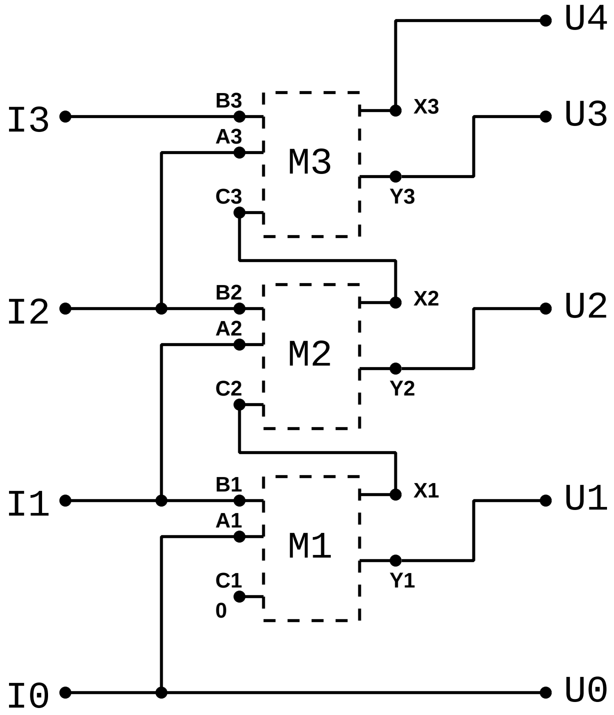
n.	A	B	C	X	Y
1	0	0	0		
2	0	0	1		
3	0	1	0		
4	0	1	1		
5	1	0	1		
6	1	1	1		
7	1	1	0		
8	1	0	0		

(c) (punti: 3)
Disegnare la rappresentazione BDD canonica delle funzioni X e Y di uscita, assumendo l'ordinamento delle variabili: prima A, poi B e poi C.

(d) (punti: 3)
Ridurre il circuito togliendo le componenti ridondanti, in modo da usare il minor numero possibile di funzioni elementari a due ingressi

(e) (punti: 4)

Considerare la seguente interconnessione di 3 moduli circuitali, ed interpretando gli insiemi di variabili booleane $I3, I2, I1, I0$ e $U4, U3, U2, U1, U0$ come rappresentazioni binarie di numeri naturali, dire quale funzione viene calcolata:



(f) (punti: 7)

completare la seguente descrizione in linguaggio VHDL del circuito a 3 ingressi e 2 uscite definito al punto (d):

```
entity M is
 port (A,B,C: in bit; X,Y: out bit);
end ha;
library lib; use lib.all;
architecture M_st of M is
component andg
    port (in1, in2 : bit; out1 : out bit);
end component;
        for all:andg use entity lib.andg;
component org
    port (in1, in2 : bit; out1 : out bit);
        for all:org use entity lib.org;
end component;
component invg
    port (in1 : bit; out1 : out bit);
        for all:invg use entity lib.invg;
end component;
component xnorg
    port (in1, in2 : bit; out1 : out bit);
        for all:xnorg use entity lib.xnorg;
end component;

        signal ...
```

```
begin
  ...



end M_st;
```

7. (punti: 14)
   Considerare la seguente funzione "`ff`" definita in Assembler SPARC:

```
ff:     save %sp,-112,%sp
        ld [%i1],%l2
        cmp %l2,0
        bge,a .LL2
        ld [%i3],%l0
        b .LL22
        st %i0,[%i1]
.LL2:   cmp %i0,%l2
        blu .LL3
        ld [%i2],%l4
        cmp %l4,0
        sll %l4,2,%g4
        be .LL5
        add %i4,%g4,%g4
        ld [%g4],%l1
        cmp %i0,%l1
        bgeu .LL4
        mov %g4,%g3
.LL5:   st %i0,[%g4+4]
        add %g4,4,%g4
        b .LL6
        add %l4,1,%l4
.LL4:   cmp %l4,1
        bleu .LL8
        mov %l4,%g2
.LL11:  ld [%g3-4],%g1
        cmp %g1,%i0
        bgeu .LL8
        add %g2,-1,%g2
        st %g1,[%g3]
        cmp %g2,1
        bgu .LL11
        add %g3,-4,%g3
.LL8:   st %i0,[%g3]
        add %l4,1,%l4
        st %l1,[%g4+4]
        add %g4,4,%g4
.LL6:   add %l0,1,%l0
        cmp %l4,%l0
        bleu .LL12
```

```
        sll %l0,2,%g3
        ld [%g4],%l1
        st %l2,[%i5+%g3]
        st %l1,[%i1]
        st %l0,[%i3]
        b .LL22
        mov %l1,%i0
.LL12:  b .LL23
        st %l4,[%i2]
.LL3:
        cmp %l0,0
        sll %l0,2,%g4
        be .LL14
        add %i5,%g4,%g4
        ld [%g4],%l1
        cmp %i0,%l1
        bleu .LL13
        mov %g4,%g3
.LL14:  st %i0,[%g4+4]
        add %g4,4,%g4
        b .LL15
        add %l0,1,%l0
.LL13:  cmp %l0,1
        bleu .LL17
        mov %l0,%g2
.LL20:  ld [%g3-4],%g1
        cmp %g1,%i0
        bleu,a .LL17
        add %g2,-1,%g2
        st %g1,[%g3]
        cmp %g2,1
        bgu .LL20
        add %g3,-4,%g3
.LL17:  st %i0,[%g3]
        add %l0,1,%l0
        st %l1,[%g4+4]
        add %g4,4,%g4
.LL15:  add %l4,1,%l4
        cmp %l0,%l4
        bleu .LL21
        sll %l4,2,%g3
        ld [%g4],%l1
        st %l2,[%i4+%g3]
        st %l1,[%i1]
        st %l4,[%i2]
        b .LL22
        mov %l1,%i0
.LL21:  st %l0,[%i3]
.LL23:  mov %l2,%i0
.LL22:  ret
```

```
        restore
```

`ff` viene chiamata da un programma principale il quale inserisce sei parametri nei registri `%o0, %o1, %o2, %o3, %o4, %o5`:
`%o4, %o5` contengono gli indirizzi dei primi elementi di due vettori di interi di almeno 3 elementi;
`%o2, %o3` contengono gli indirizzi di due variabili intere, inizializzate entrambe al valore 0 prima della prima chiamata della funzione `ff`, ed il cui valore rimane inalterato tra una chiamata e l'altra di `ff`.
`%o1` contiene l'indirizzo di una variabile intera, inizializzata al valore -1 prima della prima chiamata della funzione `ff`, ed il cui valore rimane inalterato tra una chiamata e l'altra di `ff`.
La funzione `ff` ritorna un valore intero nel registro `%o0` al programma chiamante.
Supponendo che `ff` venga quindi richiamata per 7 volte consecutive, ogni volta passando nel registro `%o0` il valore di una cifra del vostro numero di matricola, scrivere la sequenza dei 7 valori interi calcolati dalla funzione.

8. (punti: 12)
Completare la seguente traduzione della prima parte della funzione `ff` della domanda precedente (fino all'etichetta `LL3` del codice SPARC) in assembler VM-2 con le 4 istruzioni mancanti alle etichette `X1`, `X2`, `X3` e `X4`.

```
ff:    LODL 5
       PSHI
       LODL -2
X1:
       LODL 6
       STOL -2
       LODL 5
       POPI
       LODL 6
       RETN
l1:    LODL 3
       PSHI
       LODL 4
       PSHI
       LODL -2
X2:
       ADDL 6
       JNEG LL3
       LODL -4
       JZER l5
       ADDL 2
       PUSH
       PSHI
       LODL -6
       SIGN
       ADDL 6
       JNEG l6
```

```
        LODL -4
        PUSH
        LODL -5
        PUSH
        LOC8 1
X3:
        PUSH
        ADDL -7
l2:     ADDL -9
        JNEG l4
        LODL -8
        ADDL -9
        PSHI
        LODL 6
        SIGN
        ADDL -10
        JPOS l3
        LODL -8
X4:
        ADDL -9
        STOL -8
        LODL -7
        ADDL -9
        STOL -7
        ADDL -9
        JUMP l2
l3:     INSP 1
l4:     LODL 6
        PUSH
        LODL -8
        POPI
        INSP 3
        LODL -6
        PUSH
        LOC8 1
        ADDL -4
        STOL -4
        ADDL 2
        STOL -5
        POPI
        JUMP l7
l5:     DESP 2
l6:     LOC8 1
        ADDL -4
        STOL -4
        ADDL 2
        STOL -5
        LODL 6
        STOL -6
        PUSH
```

```
        LODL -5
        POPI
l7:     LOC8 1
        ADDL -3
        STOL -3
        SIGN
        ADDL -4
        JPOS l8
        INSP 2
        LODL 4
        POPI
        LODL -2
        RETN
l8:     LODL -2
        PUSH
        LODL -3
        ADDL 3
        POPI
        LODL -6
        PUSH
        LODL 5
        POPI
        LODL -3
        PUSH
        LODL 3
        POPI
        POP
        RETN
LL3: ... etc.
```

I parametri in ingresso a `ff` vengono in questo caso passati sullo stack, ed il valore calcolato viene ritornato la programma chiamante mediante il registro Accumulatore.

Risposte Scritto di Architettura, 23/7/1996

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

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

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

(b) (punti: 4)

n.	A	B	C	X	Y
1	0	0	0	0	0
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	0	1
5	1	0	1	1	0
6	1	1	1	1	1
7	1	1	0	1	0
8	1	0	0	0	1

(c) (punti: 3)
Disegnare la rappresentazione BDD canonica delle funzioni X e Y di uscita, assumendo l'ordinamento delle variabili: prima A, poi B e poi C.

(d) (punti: 3)
Ridurre il circuito togliendo le componenti ridondanti, in modo da usare il minor numero possibile di funzioni elementari a due ingressi

(e) (punti: 4)
$U = 1.5 \cdot I$

(f) (punti: 7):

```
 signal D,E,F,G,H:bit;
begin
 M_or1: org
  port map(in1=>B,in2=>C,out1=>D);
 M_xnor: xnorg
  port map(in1=>B,in2=>C,out1=>E);
 M_not: invg
  port map(in1=>A,out1=>F);
 M_and1: andg
  port map(in1=>A,in2=>D,out1=>X);
 M_and2: andg
  port map(in1=>F,in2=>D,out1=>G);
 M_and3: andg
  port map(in1=>A,in2=>E,out1=>H);
 M_or2: org
  port map(in1=>G,in2=>H,out1=>Y);
end M_st;
```

7. (punti: 14) Ogni volta restituisce il valore mediano tra tutti quelli inseriti fino a quel momento. Esempio: matricola= 1 3 5 9 6 4 2, valori= 1 1 3 3 5 5 4

8. (punti: 12)

```
       ...
       LODL -2
X1:    JPOS l1
       LODL 6
       ...
       LODL -2
X2:    SIGN
       ADDL 6
       ...
       LOC8 1
X3:    SIGN
       PUSH
       ...
       LODL -8
X4:    POPI
       ADDL -9
       ...
```

Risposte Scritto di Architettura, 23/7/1996

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.	A	B	C	X	Y
1	0	0	0		
2	0	0	1		
3	0	1	0		
4	0	1	1		
5	1	0	1		
6	1	1	1		
7	1	1	0		
8	1	0	0		

(c) (punti: 3)

(d) (punti: 3)

(e) (punti: 4)

(f) (punti: 7)

```
architecture M_st of M is etc.
  signal ...
begin
  ...








end M_st;
```

7. (punti: 14) matricola= , valori= ,

8. (punti: 12)

```
       ...
       LODL -2
X1:
       LODL 6
       ...
       LODL -2
X2:
       ADDL 6
       ...
       LOC8 1
X3:
       PUSH
       ...
       LODL -8
X4:
       ADDL -9
       ...
```