033276124541 014426113420 004133317526 027145310013 011734510013 002153407254 010427114341

Scritto di Architettura degli Elaboratori 22-2-99

Cognome: .... Nome: .... Login: 199...s.........

Domande

1. (punti: -1, 4)
Il "codice fiscale" assegnato a ciascun cittadino italiano é formato da 16 caratteri: 9 lettere dell'alfabeto inglese e 7 cifre decimali, ed ha la forma seguente (dove L indica una lettera e C indica una cifra): `LLL LLL CCLCC LCCCL`. Supponiamo che tutte le combinazioni siano ugualmente possibili, senza tener conto dell'attuale popolazione italiana. Codifichiamo i codici fiscali utilizzando la codifica ASCII a 7 bit di ciascuno dei sui 16 caratteri (112 bit in totale). Tale codice:

   (a) é molto ridondante (R$\geq 10^9$);

   (b) é poco ridondante (R$\leq 10^6$);

   (c) non é ridondante (R< 2);

   (d) é direttamente stampabile su un dispositivo di uscita senza bisogno di una libreria di sistema;

   (e) può essere "compresso" in una rappresentazione con meno di 112 bit senza perdita di informazione;

   (f) nessuna delle affermazioni precedenti é corretta.

2. (punti: -1, 4)
In un bus sincrono:

   (a) é necessario che il segnale di clock del bus sia lo stesso che controlla il funzionamento della CPU (per intenderci: il clock a 4 fasi della CPU nell'esempio VM1);

   (b) é necessario che il segnale di clock del bus sia lo stesso che controlla il funzionamento della RAM;

   (c) é necessario un meccanismo di "time out" per evitare che il master aspetti inutilmente una risposta che non arriva (per esempio a causa di un guasto dello slave);

   (d) é necessario che i segnali sui fili di controllo abbiano la stessa frequenza del segnale di clock del bus;

   (e) una operazione di lettura o scrittura deve esser portata a termine in un solo ciclo di clock del bus;

   (f) nessuna delle affermazioni precedenti é corretta.

3. (punti: -1, 4)
Riferendosi alle macchine VM1 e VM2 viste a lezione:

   (a) i registri IR, B e C fanno parte dell'architettura della VM2;

   (b) i registri FP ed SP fanno parte dell'architettura della VM2 e non fanno parte dell'architettura della VM1;

   (c) lo stack di sistema fa parte dell'architettura della VM2 e non fa parte dell'architettura della VM1;

   (d) i registri MAR, MBR, ACC e MIR non fanno parte dell'architettura della VM2;

(e) il clock di sistema fa parte dell'architettura della VM1 e non fa parte dell'architettura della VM2;

(f) nessuna delle affermazioni precedenti é corretta.

4. (punti: -1, 4)
Siano x e y due numeri naturali compresi tra 0 e 127. Rappresentiamo x e $-y$ in codice complemento a due su 8 bit, e indichiamo con "xx" e "yy" le due codifiche così ottenute. Ora aggiungiamo 4 zeri in testa a xx e yy (quindi nelle posizioni più significative) e diamo in input il tutto ad un addizionatore binario a 12 bit. Prendiamo l'output (a 12 bit) dell'addizionatore, togliamo via le 4 cifre di testa (le piu' significative) e chiamiamo "zz" la stringa di bit cosi' ottenuta.

(a) zz é la rappresentazione in complemento a due su 8 bit del numero $x - y$;

(b) zz può non essere la rappresentazione in complemento a due su 8 bit del numero $x-y$, e questo succede quando si verifica un overflow durante la somma;

(c) zz é solo "garbage" (cioé "spazzatura"), ossia non ha niente a che vedere con x, y o $x - y$;

(d) zz é la rappresentazione del numero $x - y$ in modulo e segno;

(e) per come abbiamo impostato il problema, non é possibile che si verifichi overflow;

(f) nessuna delle affermazioni precedenti é corretta.

5. (punti: -1, 4)
Una memoria cache da 512KB organizzata in 2 insiemi di associatività, con linee da 16 byte, gestione della scrittura di tipo "write-through", inserita in un sistema con indirizzamento della memoria virtuale a livello di singoli byte:

(a) nel caso adotti un campo "tag" di 9 bit non permette il "caching" di memoria RAM di dimensione superiore a 64MB;

(b) nel caso adotti un campo "tag" di 10 bit permette il "caching" di memoria RAM di dimensione non superiore a 256MB;

(c) consente di mantenere simultaneamente in cache i byte di indirizzo 3, 15, 255, 4095, 65535 e 65536;

(d) non può essere usata per il caching del codice dal momento che questo può essere richiesto solo in modalità fetch;

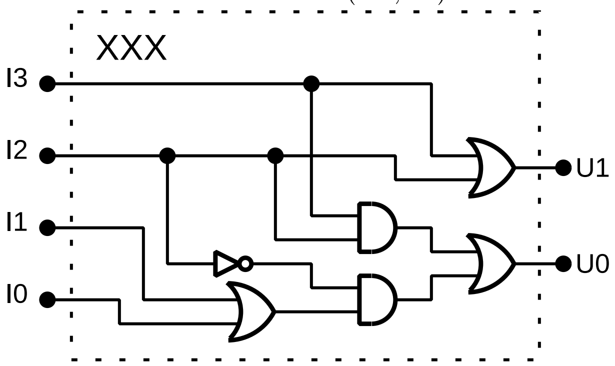
(e) se é caratterizzata da un tempo di accesso di 10ns contro i 60ns della RAM permette di ottenere un aumento della velocità della CPU per l'esecuzione delle istruzioni fino a 6 volte rispetto ad un equivalente sistema senza cache, a seconda delle maggiori o minori caratteristiche di località degli accessi;

(f) nessuna delle affermazioni precedenti é corretta.

6. (punti: 3)
Consideriamo codici binari ad espansione con tre posibili formati: 8, 12, 16 bit. Seguendo le definizioni sulle dispense, quante sono le configurazioni utili nel caso che usiamo le tecniche "con cifra di espansione" e "con configurazione di espansione"?

formato	cifra di espansione	configurazione di espansione
8 bit	...... conf. utili	...... conf. utili
12 bit	...... conf. utili	...... conf. utili
16 bit	...... conf. utili	...... conf. utili

7. Considerare il modulo circuitale XXX rappresentato in figura, caratterizzato da 4 variabili booleane di ingresso (I3,I2,I1,I0) e 2 variabili booleane di uscita (U1,U0):



(a) (punti: -1, 1)

Il circuito é realizzato in logica a 3 livelli? | si | no |

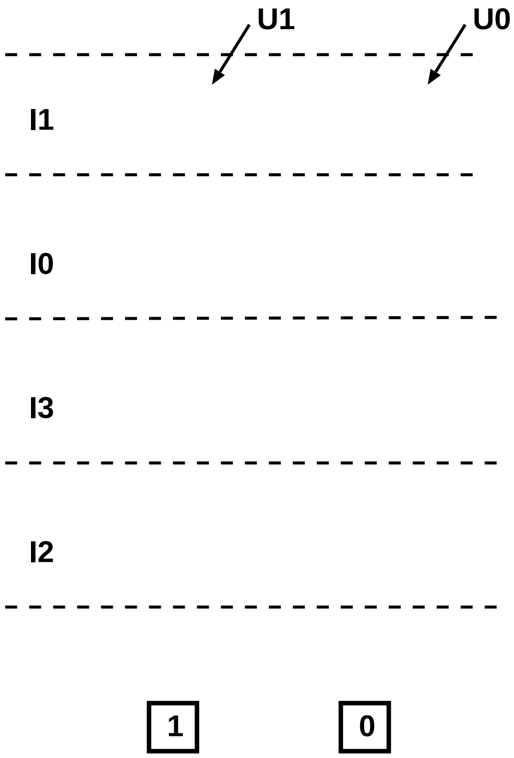
(b) (punti: 4)

riempire la seguente porzione di Tavola di Verità:

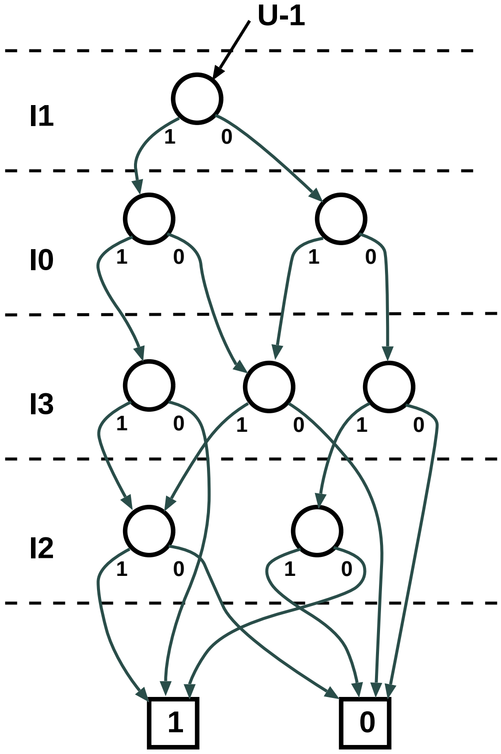
I3	I2	I1	I0	U1	U0
0	0	0	0	...	...
0	0	0	1	...	...
0	0	1	0	...	...
0	0	1	1	...	...
1	0	0	0	...	...
1	0	0	1	...	...

(c) (punti: 4)

disegnare la rappresentazione BDD canonica delle funzioni U1 ed U0, assumendo l'ordinamento per le variabili di ingresso (dall'alto al basso): I1, I0, I3, I2



8. Supponiamo di voler aggiungere al modulo circuitale XXX del punto precedente una ulteriore uscita che chiamiamo U-1, la cui funzione é definita dalla seguente rappresentazione BDD:



Chiamiamo YYY il circuito risultante a 4 ingressi e 3 uscite.

(a) (punti: -1, 1)

La rappresentazione BDD é canonica? | si | no |

(b) (punti: 4)

completare la seguente rappresentazione mediante mappa di Karnaugh della funzione U-1:

U-1		I3	0	0	1	1
I1	I0	I2	0	1	1	0
0	0					
0	1					
1	1					
1	0					

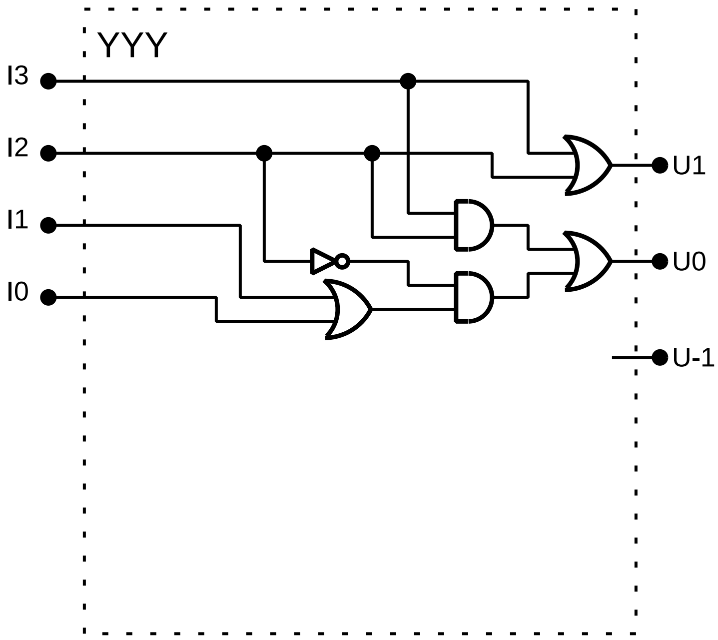
(c) (punti: 3)

Scrivere la rappresentazione in forma normale disgiuntiva (somma di prodotti) minimale per la funzione:

U-1 = ...

(d) (punti: 4)

Completare la rappresentazione grafica del circuito YYY aggiungendo a quella del circuito XXX il minor numero di elementi di tipo NOT, NAND e XNOR (é possibile realizzare la funzione aggiungendo solo 5 elementi):



(e) (punti: 3)
Assumendo che i quattro ingressi vengano interpretati come rappresentazione binaria su 4 bit di numeri naturali, indicare quale funzione matematica viene calcolata dal circuito YYY e quale codice viene usato per la rappresentazione dell'uscita.

9. (punti: 12)
Il seguente programma realizza il "bootstrap" per la macchina VM-2. Il programma prende in ingresso una successione di istruzioni VM-2 e le carica in RAM a partire dalla cella di indirizzo 90; terminato il caricamento, il bootstrap salta all'esecuzione della prima istruzione del programma caricato alla cella di indirizzo 90.

Il programma usa una procedura di lettura (che NON riportiamo): il codice della procedura inizia all'indirizzo 50; legge dall'input una istruzione (terminata da un "return"), calcola il numero binario corrispondente e lo mette sullo stack in posizione `M[FP+1]` (usando il suo FP) e lascia un valore diverso da zero nell'accumulatore; quando trova la fine della successione di istruzioni (un "return" e basta), mette zero nell'accumulatore.

Completare il programma di bootstrap con le quattro istruzioni mancanti:

```
 0:  LOC8 253
 1:  LOCH 15     (oppure LOCH 3840)
 2:  SWAS
 3:  LOC8 255
 4:  LOCH 15     (oppure LOCH 3840)
 5:
 6:  DESP 3
 7:
 8:  STOL -2
 9:  LOC8  1
10:  STOL -3
11:  CALL 50
12:  JZER 90
```

```
13:  LODL -2
14:  POPI
15:
16:  LODL -2
17:  ADDL -3
18:  STOL -2
19:
```

10. Consideriamo la realizzazione VM-1 della macchina convenzionale VM-2, e supponiamo di modificare la definizione di quest'ultima, sostituendo una nuova istruzione chiamata "WHAT yyy" alla "JNEG yyy" (ossia riutilizzando lo stesso codice operativo 0001 per specificare una serie di azioni diverse). Le nuove microistruzioni che realizzano "WHAT" sono allocate nelle celle della Control Store di indirizzo 34, 35 , 37 (e sostituiscono quelle usate per la realizzazione di "JNEG" ed una microistruzione "inutile").. La nuova istruzione usa anche le microistruzioni "vecchie" agli indirizzi 46 e 47. Riportiamo qui sotto il formato binario di rappresentazione di tutte le microistruzioni (ad eccezione del campo "Int" che comunque "non viene mai usato"):

indir. in CS	ALU	CS	RnW	MAR	MBR	D mpx	A bus	B bus	CA bus	CA en	CD bus	CD en	D bus	m cond	Addr
34	xxx	1	1	1	0	xx	11	xx	xx	0	xx	0	xx	000	xxxxxxx
35	xxx	0	1	0	1	xx	xx	xx	xx	0	xx	0	xx	001	0100101
37	000	0	x	0	0	10	xx	xx	11	1	xx	0	xx	001	0101110
46	000	1	0	1	1	01	11	xx	xx	0	xx	0	01	000	xxxxxxx
47	xxx	0	0	0	0	xx	xx	xx	xx	0	xx	0	xx	001	0000001

(a) (punti: 4)
Sapendo che, dopo aver eseguito la fase di fetch dell'istruzione "WHAT" i seguenti registri contengono i valori indicati, scrivere il contenuto degli stessi (anche in modo "simbolico", es. `M[FP+SP]`) al termine dell'esecuzione del gruppo di microistruzioni:
IR : 0001 0000 0010 0110
ACC : 17 (decimale)
ADR = 0000 0010 0110
MAR = 0000 0010 0010

(b) (punti: 3)
Quali tipi di indirizzamento utilizza l'istruzione "WHAT yyy"?

(c) (punti: 3)
Usando la notazione solita delle dispense per le istruzioni VM-2 ad esempio, per PUSH la notazione "`M[SP] <- ACC ; SP <- SP-1`") scrivere qual'Íeffetto dell'istruzione "WHAT yyy" a livello di macchina convenzionale. Se necessario aggiungere commenti a parole.

033276124541 014426113420 004133317526 027145310013 011734510013 002153407254 010427114341

Risposte scritto di Architettura degli Elaboratori, 22/2/1999

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

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

formato	cifra di espansione	configurazione di espansione
8 bit	2^7 conf. utili	2^8-1 conf. utili
12 bit	2^{10} conf. utili	2^4-1 conf. utili
16 bit	2^{14} conf. utili	2^4 conf. utili

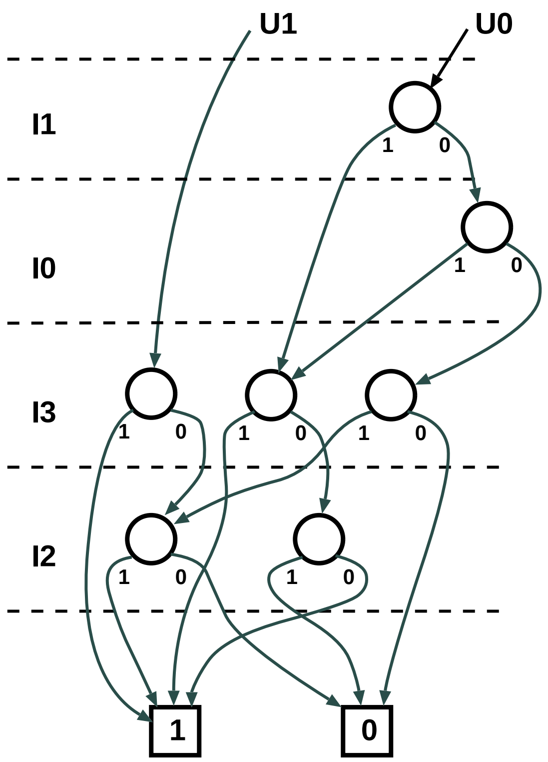
7.

(a) (punti: -1, 1) Il circuito é realizzato in logica a 3 livelli? | SI | no |

(b) (punti: 4)

I3	I2	I1	I0	U1	U0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0

(c) (punti: 4)



8. **(a)** (punti: -1, 1) La rappresentazione BDD é canonica?

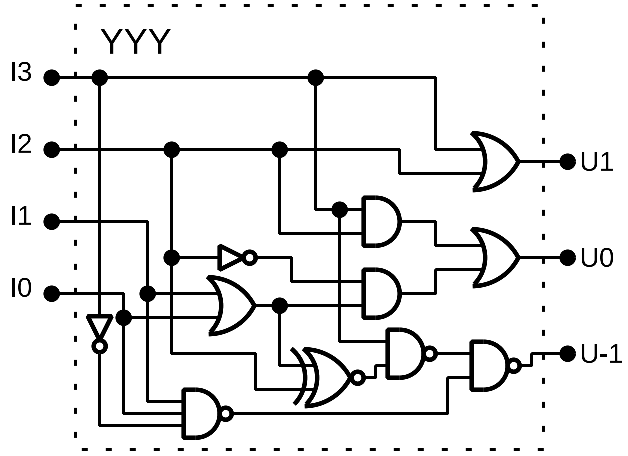
SI	no

(b) (punti: 4)

U-1		I3	0	0	1	1
I1	I0	I2	0	1	1	0
0	0		0	0	0	1
0	1		0	0	1	0
1	1		1	1	1	0
1	0		0	0	1	0

(c) (punti: 3) U-1 = I0·I1·¬I3 + I1·I2·I3 + I0·I2·I3 + ¬I0·¬I1·¬I2·I3

(d) (punti: 4)



(e) (punti: 3) Radice quadrata, con rappresentazione dell'uscita in virgola fissa, due bit per la parte intera ed un bit per la parte frazionaria.

9. (punti: 12)

```
 5:  SWAF
 7:  LOC8 90
15:  DESP 1 oppure PUSH
19:  JUMP 11
```

10.

(a) (punti: 4)

IR =	immutato	ACC =	immutato
ADR =	M[38]	MAR =	M[38]

(b) (punti: 3) Registro (implicito) e Indiretto mediante cella di memoria.

(c) (punti: 3) `M[M[yyy]]`←`ACC`, ossia l'istruzione é una "STORE Indiretta".