025276241541 014246131420 021743510130 011734501031 004133317526 001523407254 010427114341

Scritto di Architettura degli Elaboratori 4-6-99

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

Domande

1. (punti: -1, 4)
   Il codice "base64" viene utilizzato per inviare informazioni arbitrarie via posta elettronica. L'informazione viene codifica in base 64 (cioé usando un alfabeto di 64 simboli). In particolare, viene usato come alfabeto un sottoinsieme di 64 caratteri "stampabili" del codice ASCII (ciascuno espresso su 7 bit, anziché sui 6 che sarebbero strettamente necessari). Tale codice:

   (a) é sempre ridondante (R$\geq$ 2);

   (b) é poco ridondante (R$<$ 4);

   (c) permette di stampare direttamente il valore codificato senza bisogno di decodifica;

   (d) permette di codificare una parola della macchina convenzionale VM-2 (16 bit) usando 3 caratteri con ridondanza R= 2^5;

   (e) permette di codificare una parola della macchina convenzionale VM-2 (16 bit) usando 3 caratteri con ridondanza R= 2^3;

   (f) nessuna delle affermazioni precedenti é corretta.

2. (punti: -1, 4)
   In un bus asincrono si utilizzano protocolli di coordinamento tra Master e Slave (per esempio handshaking a due variabili):

   (a) per evitare che due o più master possano contemporaneamente prendere il controllo del bus;

   (b) per semplificare la parte hardware del bus eliminando i fili di controllo;

   (c) per permettere a master e slave di sapere quando inviare informazioni e quando aspettare;

   (d) per evitare conflitti tra slave sul bus;

   (e) per adattarsi alle diverse velocità di risposta dei dispositivi connessi al bus;

   (f) nessuna delle affermazioni precedenti é corretta.

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

   (a) Il microprogramma contenuto nella Control Store é un traduttore per il linguaggio della VM2;

   (b) Il microprogramma contenuto nella Control Store é un interprete scritto nel linguaggio di VM1 per il linguaggio della VM2;

   (c) il contenuto della Control Store non é propriamente un microprogramma, ma solo una rappresentazione binaria di dati usati da VM1 per realizzare VM2;

   (d) ogni istruzione del linguaggio di VM2 richiede almeno 2 cicli completi di Fetch per la sua esecuzione;

   (e) ogni microistruzione richiede 1 ciclo completo di clock (4 fasi) per la sua esecuzione;

   (f) nessuna delle affermazioni precedenti é corretta.

4. (punti: -1, 4)
La codifica dei numeri razionali secondo la tecnica "virgola mobile" con normalizzazione (esempio: standard IEEE):

   (a) rispetto alla tecnica "virgola fissa" ha il vantaggio di semplificare l'operazione di somma algebrica;

   (b) rispetto alla tecnica "virgola fissa" e a parità di numero di bit usati, permette di rappresentare un maggior numero di valori diversi;

   (c) viene usata perché permette di distinguere i razionali dai numeri interi;

   (d) viene usata perché permette di rappresentare numeri molto grandi e molto piccoli (in valore assoluto) utilizzando un numero di bit relativamente contenuto;

   (e) proprio per effetto della normalizzazione garantisce l'unicità della rappresentazione di ogni numero rappresentabile;

   (f) nessuna delle affermazioni precedenti é corretta.

5. (punti: -1, 4)
Una memoria cache migliora l'accesso da parte del processore al codice ed ai dati presenti in memoria:

   (a) sfruttando le caratteristiche di località nel tempo ed il maggior tempo di accesso dei dispositivi di memoria dinamica della Cache rispetto alla RAM;

   (b) sfruttando le caratteristiche di località nello spazio e la minore dimensione della Cache rispetto alla RAM;

   (c) sfruttando la ricerca associativa ed il miglior rapporto prestazioni/costo della Cache rispetto alla RAM;

   (d) introducendo protocolli di coerenza dei dati utili a coordinare l'attività della CPU e quella dei dispositivi DMA;

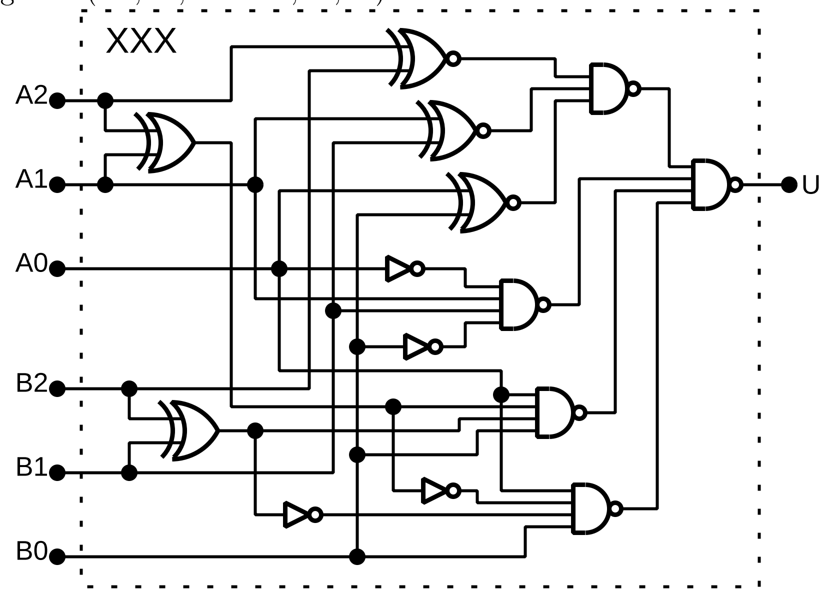
   (e) consentendo di usare bus di maggiori dimensioni maggiori (più fili in parallelo) rispetto alla connessione diretta tra RAM e CPU, adatti a supportare processori multiscalari;

   (f) nessuna delle affermazioni precedenti é corretta.

6. (punti: 3)
Supponiamo di voler aggiungere un meccanismo di memoria virtuale a segmentazione alla macchina VM-2, senza cambiare la codifica delle istruzioni e l'organizzazione fisica della RAM. Volendo scegliere tra segmentazione implicita ed esplicita, calcolare il numero di bit di indirizzamento da usare e la massima dimensione dei segmenti, supponendo di voler introdurre tre segmenti separati per codice, dati e stack.

tipo di segm.	bit per numero di segmento	max dimensione segm.
segm. esplicita	......	......
segm. implicita	......	......

7. Considerare il modulo circuitale XXX rappresentato in figura, caratterizzato da due terne di variabili booleane di ingresso (A2,A1,A0 e B2,B1,B0) ed una variabile booleana di uscita (U):

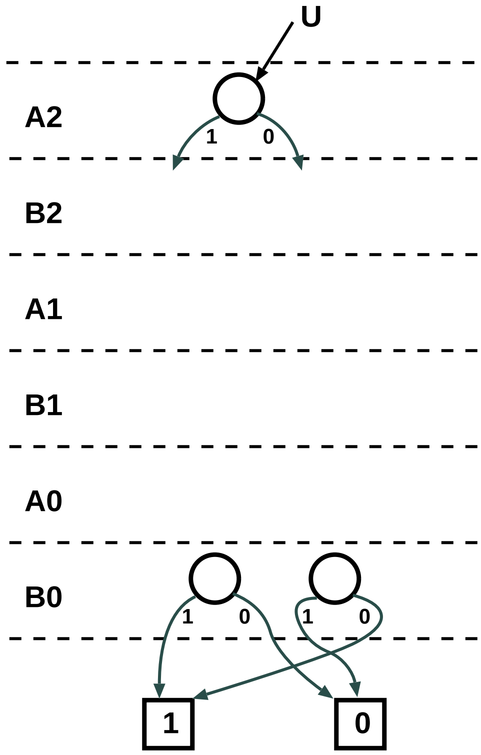


(a) (punti: -1, 1)
Il circuito é realizzato in logica a 3 livelli? | si | no |

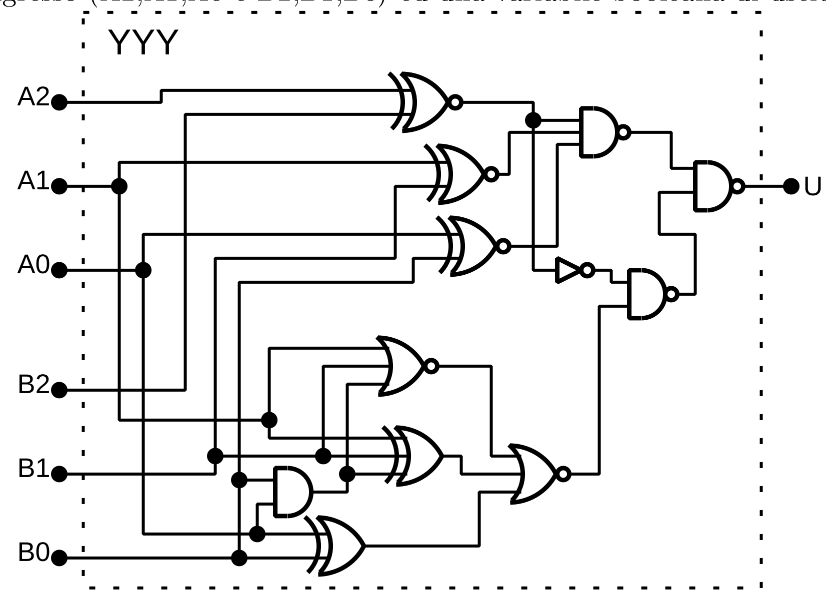
(b) (punti: 4) riempire la seguente porzione di Tavola di Verità:

A2	A1	A0	B2	B1	B0	U
0	0	0	0	0	0	...
0	0	1	0	0	0	...
0	0	1	0	0	1	...
0	0	1	0	1	0	...
0	0	1	0	1	1	...
0	0	1	1	0	0	...
0	0	1	1	0	1	...
0	0	1	1	1	0	...
0	0	1	1	1	1	...
1	1	1	1	1	1	...

(c) (punti: 4) disegnare la rappresentazione BDD canonica della funzione U, assumendo l'ordinamento per le variabili di ingresso (dall'alto al basso): A2, B2, A1, B1, A0, B0



8. Considerare il modulo circuitale YYY rappresentato in figura, caratterizzato da due terne di variabili booleane di ingresso (A2,A1,A0 e B2,B1,B0) ed una variabile booleana di uscita (U):



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

combin.	asincr.	sincrono

(b) (punti: 8)
completare la seguente rappresentazione mediante mappa di Karnaugh della funzione U:

	U		A2	0	0	0	0	1	1	1	1
			A1	0	0	1	1	1	1	0	0
B2	B1	B0	A0	0	1	1	0	0	1	1	0
0	0	0									
0	0	1									
0	1	1									
0	1	0									
1	1	0									
1	1	1									
1	0	1									
1	0	0									

(c) (punti: 4)
Scrivere la rappresentazione in forma normale disgiuntiva (somma di prodotti) minimale per la funzione:
U = ...

(d) (punti: 2)
Assumendo che le due terne di ingressi vengano interpretati come rappresentazioni di numeri con segno, indicare quale funzione viene calcolata dal circuito YYY e quale codice viene usato per la rappresentazione degli ingressi e dell'uscita.

9. (punti: 12)
Il seguente programma controlla il bilanciamento delle parentesi tonde in una successione di caratteri data in input, ignorando tutti i caratteri che non sono parentesi tonde; la fine dell'input é segnata da "#"; l'output vale 1 se le parentesi sono bilanciate, altrimenti vale 0.

Esempi:

input	output	commenti
`a++(% ( ) [ ( c) 4 ) +#`	1	la par. quadra non conta
`( ) + (ab ( f5) ;;#`	0	infatti manca una par. chiusa
`( ) + (ab ) f5) ;;#`	0	infatti manca una par. aperta

Idea dell'algoritmo:

- carico sullo stack "un po' di cose utili";
- metto "#" (o meglio, il suo codice ASCII, o meglio ...) sulla cima dello stack;
- leggo l'input, carattere per carattere; quando ho letto il carattere corrente `ch`:

  `ch` é "(": metto ")" sulla cima dello stack e leggo il carattere seguente;

  `ch` é ")": se sulla cima dello stack c'é proprio ")" allora lo tolgo (dallo stack) e continuo a leggere; altrimenti la stringa non é bilanciata: output 0 e halt;

  `ch` é "#": se in cima allo stack trovo "#" allora la stringa letta é bilanciata (output 1 e halt), altrimenti non lo é (output 0 e halt);

  `ch` é un qualunque altro carattere: leggo il carattere seguente.

Nel codice che segue, i numeri (253, 15, 255, 40, ...) sono in notazione decimale. Completare il programma con le 4 istruzioni mancanti agli indirizzi 15, 26, 46 e 47:

```
 0:  LOC8 253
 1:  LOCH 15    (oppure LOCH 3840, come su dispense)
 2:  SWAS
 3:  LOC8 255
 4:  LOCH 15    (oppure LOCH 3840)
 5:  SWAF
 6:  LOC8 40
 7:  SIGN
 8:  PUSH
 9:  LOC8 41
10:  SIGN
11:  PUSH
12:  LOC8 35
13:  SIGN
14:  PUSH
15:
16:  PUSH
17:  LODD 4094
18:  JNEG 17
19:  STOL -5
20:  ADDL -4
21:  JZER 29
22:  LODL -5
23:  ADDL -2
24:  JZER 35
25:  LODL -5
26:
27:  JZER 38
28:  JUMP 17
29:  POP
30:  ADDL -5
31:  JZER 42
32:  LOC8 48
33:  CALL  45
34:  HALT
35:  LODL -3
36:  PUSH
37:  JUMP 17
38:  POP
39:  ADDL -5
40:  JZER 17
41:  JUMP  32
42:  LOC8 49
43:  CALL 45
44:  HALT
45:  PUSH
46:
47:
48:  POP
49:  STOD 4095
50:  RETN
```

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" al posto della "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 57 (e sostituiscono quelle usate per la realizzazione di "JNEG" e due microistruzioni "inutili"). La nuova istruzione usa anche la microistruzioni "vecchia" all'indirizzo 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	110	0	x	0	0	00	01	00	11	1	xx	0	xx	000	xxxxxxx
35	xxx	1	1	1	0	xx	11	xx	xx	0	xx	0	xx	001	0100101
37	xxx	0	1	0	1	xx	xx	xx	xx	0	xx	0	xx	001	0111001
47	xxx	0	0	0	0	xx	xx	xx	xx	0	xx	0	xx	001	0000001
57	100	1	0	0	1	10	xx	xx	xx	0	xx	0	xx	001	0101111

(a) (punti: 4)

Sapendo che, dopo aver eseguito la fase di fetch dell'istruzione "WHAT":

il registro IR contiene 0001 0000 0000 1010

i registri ACC, ADR, MBR ed SP contengono i valori sotto indicati

tutte le celle dello stack contengono il numero della vostra login (quello indicato dalle ultime tre cifre decimali che compongono il nome)

scrivere il contenuto dei registri ACC, ADR, MBR ed SP (anche in modo "simbolico", es. `M[FP+SP]`) al termine dell'esecuzione del gruppo di microistruzioni:

ACC : 17 (decimale)

ADR = 0000 0000 0110

MBR = 0001 0000 0000 0110

SP = 1111 1111 0000

(b) (punti: 3)

Quali modi 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'é l'effetto dell'istruzione "WHAT yyy" a livello di macchina convenzionale. Se necessario aggiungere commenti a parole.

025276241541 014246131420 021743510130 011734501031 004133317526 001523407254 010427114341

Risposte scritto di Architettura degli Elaboratori, 4/6/1999

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

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

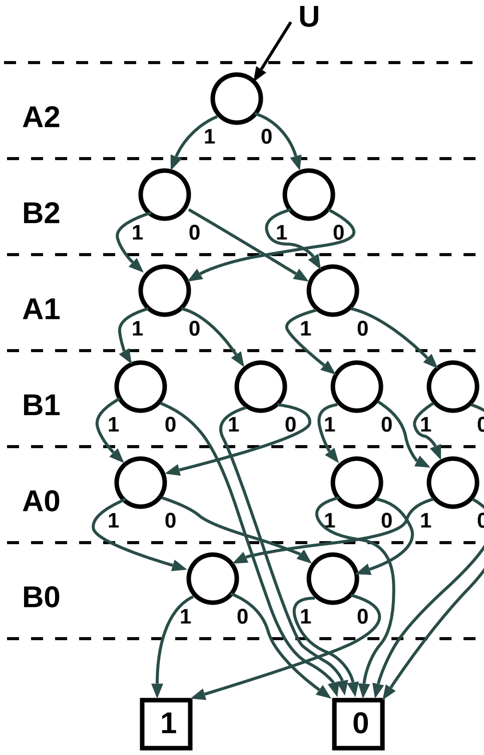
tipo di segm.	bit per numero di segmento	max dimensione segm.
segm. esplicita	2	1024
segm. implicita	0	4096

7.

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

(b) (punti: 4)

A2	A1	A0	B2	B1	B0	U
0	0	0	0	0	0	1
0	0	1	0	0	0	0
0	0	1	0	0	1	1
0	0	1	0	1	0	0
0	0	1	0	1	1	0
0	0	1	1	0	0	0
0	0	1	1	0	1	0
0	0	1	1	1	0	0
0	0	1	1	1	1	1
1	1	1	1	1	1	1



(c) (punti: 4)

8. **(a)** (punti: -1, 1) Il circuito é Combinatorio, sequenziale Asincrono o sequenziale Sincrono?

COMBIN.	asincr.	sincrono

(b) (punti: 8)

	U		A2	0	0	0	0	1	1	1	1
			A1	0	0	1	1	1	1	0	0
B2	B1	B0	A0	0	1	1	0	0	1	1	0
0	0	0		1							
0	0	1			1				1		
0	1	1				1				1	
0	1	0					1	1			
1	1	0					1	1			
1	1	1			1				1		
1	0	1				1				1	
1	0	0									1

(c) (punti: 4)

$$U = A1\cdot\neg A0\cdot B1\cdot\neg B0 + \neg A2\cdot\neg A1\cdot\neg A0\cdot\neg B2\cdot\neg B1\cdot\neg B0 + \neg A2\cdot\neg A1\cdot A0\cdot\neg B2\cdot\neg B1\cdot B0$$
$$+ \neg A2\cdot\neg A1\cdot A0\cdot B2\cdot B1\cdot B0 + \neg A2\cdot A1\cdot A0\cdot\neg B2\cdot B1\cdot B0 + \neg A2\cdot A1\cdot A0\cdot B2\cdot\neg B1\cdot B0$$
$$+ A2\cdot A1\cdot A0\cdot\neg B2\cdot\neg B1\cdot B0 + A2\cdot A1\cdot A0\cdot B2\cdot B1\cdot B0 + A2\cdot\neg A1\cdot A0\cdot\neg B2\cdot B1\cdot B0$$
$$+ A2\cdot\neg A1\cdot A0\cdot B2\cdot\neg B1\cdot B0 + A2\cdot\neg A1\cdot\neg A0\cdot B2\cdot\neg B1\cdot\neg B0$$

(d) (punti: 2) Confronta il modulo di A col modulo di B e ne indica l'uguaglianza. In uscita 0 significa falso e 1 significa vero. In ingresso i numeri sono rappresentati in complemento a 2.

9. (punti: 12)

```
15:  DESP 1  oppure  PUSH
26:  ADDL -3
46:  LODD 4095
47:  JZER 46
```

10.

(a) (punti: 4)

ACC =	17 (immutato)	ADR =	SP+10 = 4090
MBR =	N.login + 1	SP =	1111 1111 0000 (immutato)

(b) (punti: 3) Indicizzato rispetto ad SP

(c) (punti: 3) `M[SP+yyy]`$\leftarrow$`M[SP+yyy]+1`, ossia l'istruzione é un "INCR Indicizzato sullo stack top".