Esercizi per Calcolatori Elettronici

Università di Siena

Anno Accademico 2004/05

Enrico Bini

12 ottobre 2007

Esercizi

1. Dato un numero naturale X rappresentato su 4 bit (quindi X è da 0 a 15) e chiamati x_3, x_2, x_1, x_0 i bit della sua rappresentazione binaria, dal più significativo al meno significativo, trovare l'espressione dei segnali:
   - z_1 è uguale a 1 se X è pari, 0 altrimenti;
   - z_2 è uguale a 1 se X è multiplo di 3, 0 altrimenti;
   - z_3 è uguale a 1 se X è multiplo di 5, 0 altrimenti;
   - $z_4 = z_2 + z_3$;
   - z_5 è uguale a 1 se X è numero primo (0 è numero primo), 0 altrimenti;
   - $z_6 = \overline{z_5}$.
2. È noto che un decoder a N ingressi (e quindi 2^N uscite) imposta l'uscita:
   $$z_i = \begin{cases} 1 & \text{se } i = X \\ 0 & \text{se } i \neq X \end{cases}$$
   dove X è il numero intero rappresentato dagli N bit di ingresso $x_0, \ldots, x_{N-1}$. Come si realizza la rete $\overline{z_i}$, ovvero che imposta l'uscita selezionata dall'ingresso X a 0, e tutte le altre (non selezionate) a 1 (nota: questo dispositivo si chiama decoder *attivo basso*)? Come si realizza un multiplexer a partire dal decoder attivo basso?
3. È noto che un encoder a N bit:
   - riceve in ingresso 2^N bit $(x_0, x_1, \ldots, x_{2^N-1})$;
   - imposta l'uscita Z, rappresentata su N bit, tale che:
     $$Z = \max\{i : x_i = 1\}.$$

   Si realizza la rete logica composta da un decoder a N bit, le cui uscite vengono collegate agli ingressi di un encoder a N bit (vedi Figura 1). In che modo Z è legato a X?
4. Si collega in decoder in cascata a un encoder (vedi Figura 2). Il bit di validità v, in uscita dall'encoder, viene collegato in AND alle uscite del decoder. In che modo Z è legato a X?
5. Realizzare un multiplexer esclusivamente con porte NAND.
6. Realizzare il circuito di lookahead nel caso di ingressi (due) a due bit, nel caso di carry in ingresso $c_i = 0$ (rete logica con 4 ingressi e una uscita).

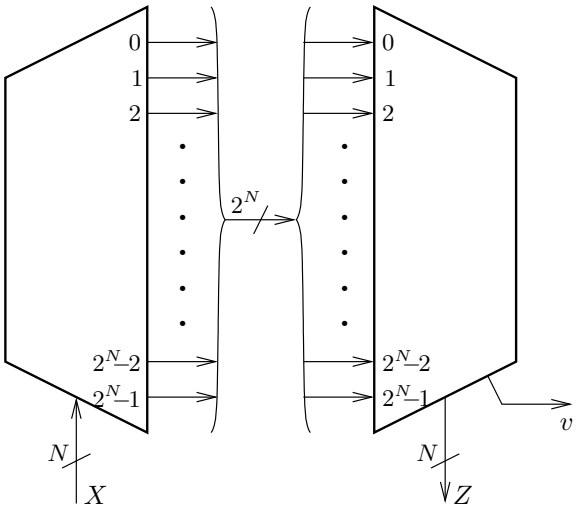

Figura 1: Encoder in cascata a decoder.

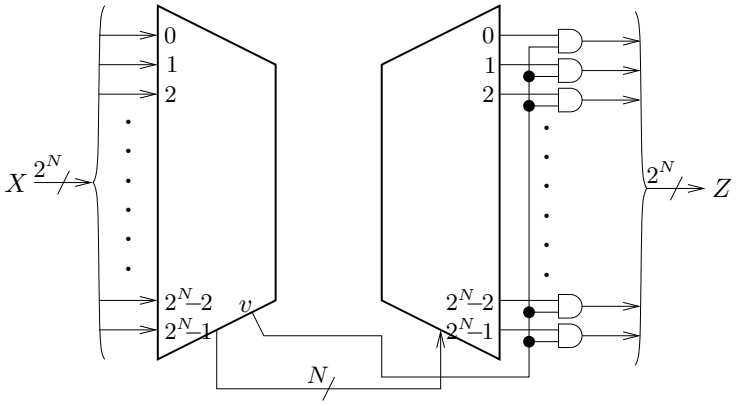

Figura 2: Decoder in cascata a encoder.

7. Realizzare il circuito di lookahead nel caso di ingressi (due) a due bit, nel caso di carry in ingresso $c_i = 1$ (rete logica con 4 ingressi e una uscita).

8. Realizzare il circuito di lookahead nel caso di ingressi (due) a due bit (rete logica con 5 ingressi e una uscita).

9. Realizzare la funzione z_4 dell'esercizio 1 usando esclusivamente porte XOR.

10. Dati due numeri naturali X e Y, entrambi rappresentati su 2 bit, realizzare la rete logica "prodotto" che produce in uscita:

    - il numero naturale Z, rappresentato su 3 bit, tale che $Z = X\,Y$;
    - un quarto bit z_v di invalidità che vale 1 quando il prodotto $X\,Y$ non è rappresentabile su 3 bit, 0 quando il prodotto è rappresentabile su 3 bit e, conseguentemente, il valore di Z ha significato.

    Una volta realizzata la rete, quale è il valore di Z quando $X = 3$ e $Y = 3$?

11. Dato un numero naturale X rappresentato su 4 bit realizzare la rete che produce il quoziente della divisione intera $X/4$ ovvero il numero $Q = \lfloor X/4 \rfloor$. Quanti bit sono necessari per la rappresentazione di Q? Realizzare la rete che produce il resto R della stessa divisione, ovvero $R = X - \lfloor X/4 \rfloor\, 4$? Quanti bit sono necessari per la rappresentazione di R?

12. Dato un numero naturale X rappresentato su 4 bit realizzare la rete che produce il quoziente della divisione intera $X/3$ ovvero il numero $Q = \lfloor X/3 \rfloor$. Quanti bit sono necessari per la rappresentazione di Q? Realizzare la rete che produce il resto R della stessa divisione, ovvero $R = X - \lfloor X/3 \rfloor 3$? Quanti bit sono necessari per la rappresentazione di R?

13. Un sommatore a 4 bit (realizzato **senza** circuito di lookahead) ha in ingresso i due numeri naturali $X = 3$ e $Y = 4$ e le sue uscite hanno raggiunto una situazione di stabilità ad un istante temporale $t^* < 0$. All'istante $t = 0$ il bit meno significativo di Y transisce da 0 a 1. Descrivere l'evoluzione temporale delle uscite del sommatore nelle ipotesi che:

    - ogni livello di logica si comporta, dal punto di vista temporale, come un puro ritardo di Δ, mantenendo quindi l'uscita stabile dal momento in cui un ingresso cambia al momento in cui l'uscita cambia con il nuovo valore;
    - il sommatore ad un bit è realizzato con due livelli di logica;
    - il sommatore a 4 bit è realizzato secondo lo schema classico che prevede l'uso di 4 sommatori ad un bit, in cui il bit di carry si propaga dal meno al più significativo.

    Dopo quanto tempo l'uscita Z può considerarsi stabile? In presenza di un circuito di lookahead a 2 bit, come cambia l'evoluzione temporale del segnale di uscita? Dopo quanto tempo può considerarsi stabile?

14. Dato un numero X da 1 a 12 rappresentato su 4 bit, il cui significato è un mese dell'anno, e un numero Y da 0 a 65535 rappresentato su 16 bit, il cui significato è l'anno, realizzare la rete che ritorna il numero Z di giorni del mese X nell'anno Y (si tenga conto degli anni bisestili, ma non della regola del calendario gregoriano che prevede che gli anni multipli di 100 ma non di 200 non sono bisestili). I giorni dei mesi di un anno non bisestile, da gennaio a dicembre, sono: $\mathsf{jan} = 31$, $\mathsf{feb} = 28$, $\mathsf{mar} = 31$, $\mathsf{apr} = 30$, $\mathsf{may} = 31$, $\mathsf{jun} = 30$, $\mathsf{jul} = 31$, $\mathsf{aug} = 31$, $\mathsf{sep} = 30$, $\mathsf{oct} = 31$, $\mathsf{nov} = 30$, $\mathsf{dec} = 31$.

15. Dati due numeri X e Y su 2 bit, realizzare un comparatore. Il comparatore produce in uscita:

    - un segnale z_0 che vale 1 quando X è uguale a Y, 0 altrimenti;
    - un segnale z_1 che, quando $X \neq Y$, vale 1 se $X < Y$, 0 se $X > Y$. Se $X = Y$ il valore di z_1 è indifferente.

16. Si realizzi una rete che implementa la successione di Fibonacci per i numeri rappresentabili su 8 bit. Si rammenta che la successione di Fibonacci viene inizializzata con i valori $a_0 = 0$ e $a_1 = 1$ e prosegue in base alla legge $a_i = a_{i-1} + a_{i-2}$ (quindi in pratica è $0, 1, 1, 2, 3, 5, 8, 13, \ldots$). La rete produce in uscita:

    - un numero Z su 8 bit che rappresenta il valore del numero della successione (non ci si preoccupi dell'*overflow* ovvero del caso in cui il numero della successione non sia più rappresentabile su 8 bit).

    La rete riceve in ingresso:

    - il segnale di marcatura p che fa avanzare la successione al prossimo numero impostando conseguentemente le uscite;
    - un segnale $/\mathsf{clr}$ che, quando marcato su valore 0, provvede ad inizializzare la rete allo stato iniziale.

    Si fornisca la temporizzazione dell'uscita nel momento in cui viene operata un'azione di *clear* e negli istanti successivi fino a quando $Z < 10$.

    Si possono utilizzare tutti i seguenti macroblochi: registro, registro con valore di default, contatore (con segnale di clear $/\mathsf{clr}$), sommatore di due numeri, incremento, multiplexer, demultiplexer, decoder, encoder, ROM.

17. Si realizzi una rete che, dato in ingresso $X = (x_1, x_0)$, produce un'uscita z che vale 1 quando il valore istantaneo di X è $(1, 0)$.

18. Si realizzi una rete che, dato in ingresso $X = (x_1, x_0)$, produce un'uscita z che vale 1 quando il valore di X, opportunamente marcato dal segnale p, ha presentato la sequenza $(1, 0)$ e $(0, 1)$.

19. Realizzare un contatore su 4 bit usando **esclusivamente** porte NOT e registri con segnale di clear (/clr) che marcano sulle transizioni $0 \rightarrow 1$ ($\_\uparrow\overline{\ }$) del loro segnale di marcatura.

20. Realizzare un contatore up-down su N bit. Questo dispositivo, rispetto al normale contatore, è dotato di un segnale di ingresso aggiuntivo chiamato up. Quando il segnale up, ad un istante di marcatura, vale 1 allora il contatore incrementa il proprio valore. Quando il segnale up vale 0 il contatore decrementa il proprio valore. Non si consideri l'evenienza di overflow (in cui il contarore decrementa 0 oppure incrementa $2^N - 1$).

    Si possono utilizzare tutti i seguenti macroblochi: registro, registro con valore di default, contatore (con segnale di clear /clr), sommatore di due numeri, incremento, sottrattore di due numeri, decremento, multiplexer, demultiplexer, decoder, encoder.

21. Realizzare una rete, sincronizzata su ogni transizione $1 \rightarrow 0$ del segnale di marcatura p, che produce ripetutamente la sequenza $1, 1, 0, 0$ (l'uscita quindi sarà $1, 1, 0, 0, 1, 1, 0, 0, 1, \ldots$). La rete è inoltre dotata di un segnale di inizializzazione init (attivo alto) che, quando marcato su valore 1, riporta la rete nello stato iniziale (che è quello in cui l'uscita è pari al primo 1). Mostrare la temporizzazione dell'uscita in presenza di evento di inizializzazione e durante la normale evoluzione del sistema.

    Per la realizzazione della rete si possono solo utilizzare porte logiche elementari (NOT, AND, OR), registro ad un bit, registro ad un bit con set/clear.

22. Realizzare un contatore da 0 a 4 (nel senso che i valori che produce in uscita sono $0, 1, 2, 3, 4, 0, \ldots$ e così via) con segnale di /clr (attivo basso) che riporta il contatore allo stato iniziale.

23. Dato lo schema di Figura 3, ricavare la temporizzazione dell'uscita Z in presenza degli ingressi di Figura 4, assumendo che il valore iniziale di tutti i registri a un bit è 0.

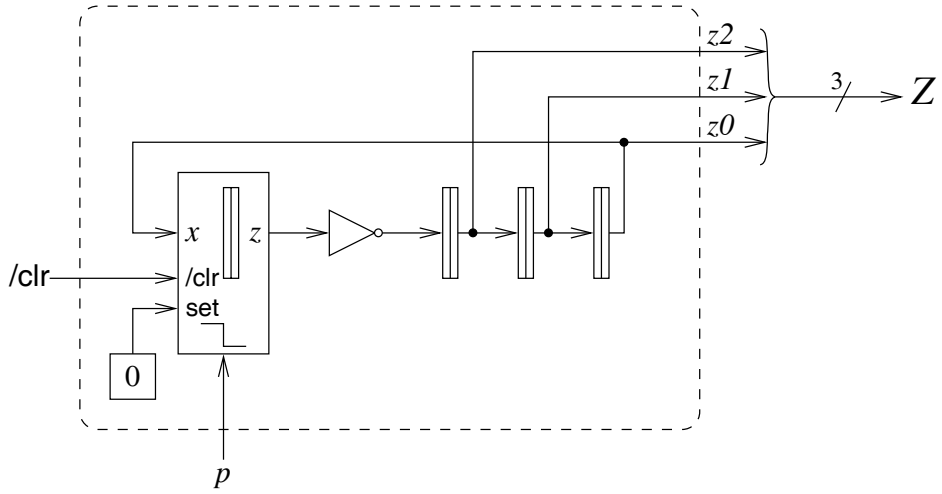

Figura 3: Schema del circuito.

24. Dato un numero intero X (quindi con segno) rappresentato in complemento a 2 su N bit, quanti bit sono necessari per rappresentare il suo modulo $|X|$? Realizzare la rete che implementa questa funzione. A tal fine si possono utilizzare i macroblocchi: sommatore, incremento, decremento, multiplexer, demultiplexer, encoder (attivo alto o basso), decoder (attivo alto o basso) e le porte logiche elementari (AND, OR, NOT).

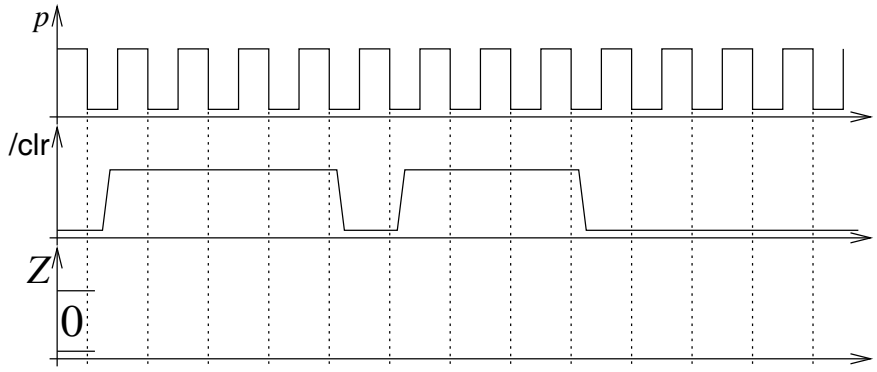

Figura 4: Temporizzazioni di ingresso.

25. Sia dato un numero X rappresentato in complemento a 2 su N. Realizzare la rete logica che produce la rappresentazione del numero X su $N + 1$ bit.

26. Come è rappresentato il numero 121 con il tipo `float` (ovvero secondo lo standard IEEE/ANSI 754 dei numeri floating point su 4 byte)? Scrivere il suo valore in esadecimale (base 16).

27. Come è rappresentato il numero $-(1/16 + 1/4)$ con il tipo `float` (ovvero secondo lo standard IEEE/ANSI 754 dei numeri floating point su 4 byte)? Scrivere il suo valore in esadecimale (base 16).

28. **Difficile.** Quale è il minimo intero positivo non rappresentabile con un numero floating point con un bit di segno, M bit di mantissa e K bit di esponente? Quanto vale questo numero nel caso di numeri `float` o `double` (secondo lo standard IEEE/ANSI 754)?

29. Realizzare una rete sequenziale sincrona dotata di un segnale /init (attivo basso) e di un'uscita Z su 8 bit. Il dispositivo produce, sugli 8 bit di uscita, la codifica ASCII di un carattere (si veda la Tabella ASCII disponibile sul sito). I caratteri prodotti ad ogni istante di marcatura, ripetono la stringa "`Ciao Zio`". Quando il segnale `/init` viene marcato su valore 0 allora il prossimo carattere sarà comunque la prima lettera "`C`", all'istante di marcatura seguente "`i`", e così via. Quando si arriva alla fine della stringa, si riparte dall'inizio.

    Per la realizzzione del dispositivo si possono utilizzare: registri, porte logiche elementari (AND, OR, NOT), encoder, decoder, multiplexer, demultiplexer, incremento, decremento e ROM.

30. Come è rappresentato il numero 125.125_{10} in **fixed point**, nelle ipotesi che abbiamo a disposizione 16 bit per la parte intera e 16 bit per la parte frazionaria? Scrivere tutta la rappresentazione in esadecimale. Quanti byte occupa? Supponiamo che tale numero viene scritto in memoria a partire dall'indirizzo `0x0C004C08`. Quali e quante locazioni di memoria sono necessarie per la sua memorizzazione? Come viene memorizzato se l'ordinamento dei byte è **big endian**? Come viene memorizzato se l'ordinamento dei byte è **little endian**?

31. Come è rappresentato il numero 125.125_{10} in **floating point**, secondo lo standard IEEE/ANSI 754 della rappresentazione dei `float`? Scrivere tutta la rappresentazione in esadecimale. Quanti byte occupa? Supponiamo che tale numero viene scritto in memoria a partire dall'indirizzo `0x0C004C08`. Quali e quante locazioni di memoria sono necessarie per la sua memorizzazione? Come viene memorizzato se l'ordinamento dei byte è **big endian**? Come viene memorizzato se l'ordinamento dei byte è **little endian**?

32. Nel linguaggio C si chiama *cast* la conversione da un formato di rappresentazione dei dati ad un altro. Per esempio, nel codice riportato sotto:

```
float varCorta;
```

```
double varLunga;

varCorta = 12.3e-10;
varLunga = (double)varCorta;  /* cast da float a double */
```

viene fatta una conversione di tipo da `float` a `double`. Realizzare una rete logica che, preso in ingresso un numero X su 32 bit interpretato come un numero `float`, produce un numero Z su 64 bit che è la rappresentazione come `double` di X.

Soluzioni

Per motivi di tempo non riesco a fornire le soluzioni complete degli esercizi proposti. Fornisco però volentieri alcune idee per la loro risoluzione e i risultati finali. In alcuni casi le soluzioni corrette sono accompagnate da ulteriori spunti di riflessione segnalati dalla parola "**Approfondimento**" in grassetto.

1. Sia $X = (x_3, x_2, x_1, x_0)$. Allora:

   - $z_1 = \overline{x_0}$;
   - $z_2 = \overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0} + \overline{x_3}\,\overline{x_2}\,x_1\,x_0 + \overline{x_3}\,x_2\,x_1\,\overline{x_0} + x_3\,\overline{x_2}\,\overline{x_1}\,x_0 + x_3\,x_2\,\overline{x_1}\,\overline{x_0} + x_3\,x_2\,x_1\,x_0$. **Approfondimento**: si può anche trovare che $z_2 = \overline{x_3 \oplus x_2}\,\overline{x_1 \oplus x_0} + (x_3 \oplus x_1)(x_2 \oplus x_0)$, che si può realizzare con 4 XOR e 5 NAND a due ingressi (piuttosto che con 4 NOT, 6 AND a quattro ingressi e 1 OR a sei ingressi).
   - $z_3 = \overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0} + \overline{x_3}\,x_2\,\overline{x_1}\,x_0 + x_3\,\overline{x_2}\,x_1\,\overline{x_0} + x_3\,x_2\,x_1\,x_0$. **Approfondimento**: si può anche trovare che $z_3 = \overline{(x_3 \oplus x_1) + (x_2 \oplus x_0)}$, che si realizza con 1 NOR e 2 XOR.
   - $z_4 = z_2 + z_3 = \overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0} + \overline{x_3}\,\overline{x_2}\,x_1\,x_0 + \overline{x_3}\,x_2\,\overline{x_1}\,x_0 + \overline{x_3}\,x_2\,x_1\,\overline{x_0} + x_3\,\overline{x_2}\,\overline{x_1}\,x_0 + x_3\,\overline{x_2}\,x_1\,\overline{x_0} + x_3\,x_2\,\overline{x_1}\,\overline{x_0} + x_3\,x_2\,x_1\,x_0$. **Approfondimento**: si può trovare che $z_4 = 1 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0$, che si realizza con 4 XOR.
   - con le mappe di Karnaugh si trova che $z_5 = \overline{x_3}\,\overline{x_2} + \overline{x_3}\,x_0 + x_2\,\overline{x_1}\,x_0 + \overline{x_2}\,x_1\,x_0$.
   - $z_6 = \overline{z_5} = (x_3 + x_2)(x_3 + \overline{x_0})(\overline{x_2} + x_1 + \overline{x_0})(x_2 + \overline{x_1} + \overline{x_0})$.

2. Il decoder attivo basso può essere ottenuto applicando la negazione alle uscite del decoder standard (attivo alto). Applicando le leggi di De Morgan, la negazione trasforma le AND in OR con gli ingressi negati. Le negazioni sugli ingressi delle OR si trasferiscono alle "linee orizzontali", negando i segnali x_i che prima non lo erano e togliendo le negazioni ai segnali che prima erano negati.

   Il multiplexer si realizza utilizzando lo stesso principio visto con il decoder attivo alto. In questo caso l'ingresso valido è selezionato da uno 0 che quindi apre le porte OR. Tutte le linee verticali in uscita dalle OR sono collegate ad una AND.

3. $Z = X$. **Approfondimento**: inoltre in segnale di uscita v vale sempre 1 perché il dato in ingresso all'encoder è sempre valido per costruzione.

4. Se $X = 0$ allora $Z = 0$. Altrimenti Z è la più grande potenza di 2 minore o uguale a X. Quindi c'è da stare attenti che $Z \neq X$. **Approfondimento**: se $X \neq 0$ allora $Z = \max\{2^i : 2^i \leq X\} = 2^{\max\{i : i \leq \log_2 X\}} = 2^{\lfloor \log_2 X \rfloor}$.

5. Si sostituisca con le porte NAND nel modo classico. Si tenga poi presente che due porte NOT collegate una dietro l'altra possono essere semplificate.

6. $c_{i+2} = x_{i+1}\,y_{i+1} + y_{i+1}\,y_i\,x_i + y_{i+1}\,x_{i+1}\,x_i$.

7. $c_{i+2} = y_{i+1}\,x_{i+1} + y_{i+1}\,x_i + y_i\,x_{i+1} + y_{i+1}\,y_i + x_{i+1}\,x_i$.

8. $c_{i+2} = y_{i+1}\,x_{i+1} + y_{i+1}\,y_i\,x_i + y_{i+1}\,x_i\,c_i + y_i\,x_{i+1}\,c_i + y_i\,x_{i+1}\,x_i + y_{i+1}\,y_i\,c_i + x_{i+1}\,x_i\,c_i$.

9. $z_4 = \overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0} + \overline{x_3}\,\overline{x_2}\,x_1\,x_0 + \overline{x_3}\,x_2\,\overline{x_1}\,x_0 + \overline{x_3}\,x_2\,x_1\,\overline{x_0} + x_3\,\overline{x_2}\,\overline{x_1}\,x_0 + x_3\,\overline{x_2}\,x_1\,\overline{x_0} + x_3\,x_2\,\overline{x_1}\,\overline{x_0} + x_3\,x_2\,x_1\,x_0 = \overline{x_3}\,\overline{x_2}\,(\overline{x_1}\,\overline{x_0} + x_1\,x_0) + \overline{x_3}\,x_2\,(\overline{x_1}\,x_0 + x_1\,\overline{x_0}) + x_3\,\overline{x_2}\,(\overline{x_1}\,x_0 + x_1\,\overline{x_0}) + x_3\,x_2\,(\overline{x_1}\,\overline{x_0} + x_1\,x_0) = \overline{x_3}\,\overline{x_2}\,(\overline{x_1 \oplus x_0}) + \overline{x_3}\,x_2\,(x_1 \oplus x_0) + x_3\,\overline{x_2}\,(x_1 \oplus x_0) + x_3\,x_2\,(\overline{x_1 \oplus x_0}) = \overline{(x_1 \oplus x_0)(\overline{x_3}\,\overline{x_2} + x_3\,x_2)} + (x_1 \oplus x_0)(\overline{x_3}\,x_2 + x_3\,\overline{x_2}) = \overline{(x_1 \oplus x_0)(\overline{x_3 \oplus x_2})} + (x_1 \oplus x_0)(x_3 \oplus x_2) = \overline{(x_1 \oplus x_0) \oplus (x_3 \oplus x_2)} = \overline{x_1 \oplus x_0 \oplus x_3 \oplus x_2} = 1 \oplus x_1 \oplus x_0 \oplus x_3 \oplus x_2$

10. Con le mappe di Karnaugh, considerando l'uscita Z *indifferente* (stato logico "$-$") quando $X = 3$ e $Y = 3$. $v = y_1\,y_0\,x_1\,x_0$. Sia $Z = \sum_{i=0}^{2} z_i 2^i$, allora

    - $z_0 = y_0\,x_0$;

- $z_1 = y_0\,x_1 + y_1\,x_0$;
- $z_2 = y_1\,x_1$.

Quando $X = 3$ e $Y = 3$ tutti gli z_i sono 1, quindi $Z = 7$ (risultato non significativo ma presente sui bit di uscita).

11. Q si rappresenta su 2 bit. R si rappresenta su 2 bit. Con mappe di Karnaugh o per ispezione diretta della tabella di verità si trova che $q_1 = x_3$, $q_0 = x_2$, $r_1 = x_1$ e $r_0 = x_0$. La conoscenza preventiva non semplifica la rete per il calcolo di R, che non è ulteriormente semplificabile.

12. Q si rappresenta su 3 bit, R su 2. Con le mappe di Karnaugh si trova che:

    - $q_2 = x_3\,x_3$;
    - $q_1 = x_3\,\overline{x_2} + \overline{x_3}\,x_2\,x_1$;
    - $q_0 = x_3\,x_1\,x_0 + x_3\,\overline{x_2}\,x_0 + x_3\,\overline{x_2}\,x_1 + \overline{x_2}\,x_1\,x_0 + \overline{x_3}\,x_2\,\overline{x_1}$;
    - $r_1 = \overline{x_3}\,x_2\,\overline{x_1}\,x_0 + x_3\,\overline{x_2}\,x_1\,x_0 + \overline{x_3}\,\overline{x_2}\,x_1\,\overline{x_0} + x_3\,\overline{x_2}\,\overline{x_1}\,\overline{x_0} + x_3\,x_2\,x_1\,\overline{x_0}$;
    - $r_0 = \overline{x_3}\,x_2\,\overline{x_1}\,\overline{x_0} + \overline{x_3}\,\overline{x_2}\,\overline{x_1}\,x_0 + x_3\,x_2\,\overline{x_1}\,x_0 + \overline{x_3}\,x_2\,x_1\,x_0 + x_3\,\overline{x_2}\,x_1\,\overline{x_0}$.

13. Si risolve tenendo conto del ritardo introdotto dal sommatore pari a 2Δ su ognuna delle sue uscite. In corrispondenza della transizione dell'ingresso y_0 da 0 a 1, l'uscita evolve così:

Intervallo	y_3	y_2	y_1	y_0	Y
$[t^*, 2\Delta)$	0	1	1	1	7
$[2\Delta, 4\Delta)$	0	1	1	0	6
$[4\Delta, 6\Delta)$	0	1	0	0	4
$[6\Delta, 8\Delta)$	0	0	0	0	0
$[8\Delta, +\infty)$	1	0	0	0	8

L'uscita può considerarsi stabile dopo 8Δ. In presenza di un circuito di lookahead a 2 bit, le uscite evolvono così:

Intervallo	y_3	y_2	y_1	y_0	Y
$[t^*, 2\Delta)$	0	1	1	1	7
$[2\Delta, 4\Delta)$	0	1	1	0	6
$[4\Delta, 6\Delta)$	0	0	0	0	0
$[6\Delta, +\infty)$	1	0	0	0	8

In questo caso l'uscita è stabile dopo 6Δ.

14. L'uscita Z, che è il numero di giorni di un mese (al massimo 31), è rappresentabile su 5 bit.

    - $z_4 = z_3 = z_2 = 1$;
    - $z_1 = x_3 + x_2 + \overline{x_1} + x_0$;
    - $z_0 = \overline{x_3}\,x_0 + x_3\,\overline{x_0} + \overline{y_1}\,\overline{y_0}\,\overline{x_3}\,\overline{x_2}$.

15. Con mappe di Karnaugh:

    - $z_0 = \overline{y_1}\,\overline{y_0}\,\overline{x_1}\,\overline{x_0} + \overline{y_1}\,y_0\,\overline{x_1}\,x_0 + y_1\,y_0\,x_1\,x_0 + y_1\,\overline{y_0}\,x_1\,\overline{x_0} = \overline{(y_1 \oplus x_1) + (y_0 \oplus x_0)}$;
    - $z_1 = y_1\,y_0 + \overline{x_1}\,\overline{x_0} + y_1\,\overline{x_1}$.

16. Una soluzione semplice ed efficace consiste nel mantenere due registri che chiamo `R0` e `R1`. Siano IN(`R`) l'ingresso del registro `R` e OUT(`R`) la sua uscita. Allora basta collegare in questo modo:

    - $Z =$ OUT(`R1`);
    - IN(`R1`) = OUT(`R0`);

- $\mathsf{IN}(\mathtt{R0}) = \mathsf{OUT}(\mathtt{R0}) + \mathsf{OUT}(\mathtt{R1})$ (in questo caso si intende somma algebrica non OR).

Il registro `R0` deve essere con valore di default pari a 1. Il segnale di ingresso /clr si collega al clear del registro `R1` e, dopo essere stato negato, al set del registro `R0`.

Approfondimento: una soluzione alternativa è possibile utilizzando un solo registro per mantenere lo stato che funge da indice in una ROM.

17. La rete è combinatoria. $z = x_1\,\overline{x_0}$.

18. Si realizza una rete sequenziale i cui stati sono 3 rappresentati sui 2 bit (y_1, y_0), mantenuti da un registro interno. Nelle ipotesi che gli stati siano così codificati:

    - $(0,0)$ è lo stato in cui nessun dato valido è stato presentato;
    - $(0,1)$ è lo stato in cui l'ultimo ingresso marcato è $(1,0)$;
    - $(1,0)$ è lo stato in cui passo quando si è presentata in ingresso la sequenza $(1,0)$ e $(0,1)$;

    l'evoluzione degli stati è data da:

    - $y_0^{(i+1)} = x_1^{(i)}\,\overline{x_0^{(i)}}$;
    - $y_1^{(i+1)} = y_0^{(i)}\,\overline{x_1^{(i)}}\,x_0^{(i)}$;
    - $z = y_1$.

19. La differenza rispetto allo schema visto a lezione è che il segnale di marcatura di ogni registro di bit viene prelevato **dopo** la porta NOT e non direttamente sull'uscita del registro.

20. Rispetto ad un normale contatore, in ingresso al registro non c'è più il valore corrente incrementato. Bensì c'è un multiplexer comandato dal bit up. Quando up vale 1 allora viene selezionato il dato incrementato. Quando up vale 0 allora il multiplexer seleziona il valore corrente decrementato, che sarà opportunamente prodotto dalla relativa rete combinatoria (disponibile fra i macroblocchi).

21. Una rapida soluzione consiste nell'osservare che:

    - lo stato può essere costituo dall'indice del bit all'interno della sequenza $1, 1, 0, 0$;
    - con tale assunzione la rete, quando si trova nello stato S_i, passa sempre allo stato S_{i+1}. Quando in S_3 passa in S_0;
    - questa legge di transizione fra stati equivale a quella del contatore a 2 bit.

    Quindi, chiamate (y_1, y_0) le due uscite di un contatore a 2 bit, l'uscita z della rete è $\overline{y_1}$.

22. Si realizza con un contatore a 3 bit che però, quando il valore corrente è 4, imposta il prossimo valore di nuovo a 0 forzatamente.

23. Per la temporizzazione di veda Figura 5.

24. I numeri rappresentabili su N bit vanno da -2^{N-1} a $2^{N-1} - 1$. Il massimo modulo è quindi 2^{N-1} che si rappresenta su N bit con $\mathtt{1}\underbrace{\mathtt{0000}\cdots\mathtt{0}}_{N-1}$.

    La funzione modulo è la seguente:

    $$|X| = \begin{cases} X & \text{se } X \geq 0 \\ -X & \text{se } X < 0 \end{cases}$$

    Siccome la rappresentazione di $-X$ è data da $\overline{X} + 1$, allora una semplice implementazione della rete che produce il modulo $|X|$ è:

    $$x_{N-1} \cdot \mathtt{inc}(\overline{X}) + \overline{x_{N-1}} \cdot X$$

    dove `inc` è la rete combinatoria di incremento. L'espressione si giustifica con il fatto che il bit x_{N-1} vale 1 quando il numero X è negativo, 0 altrimenti.

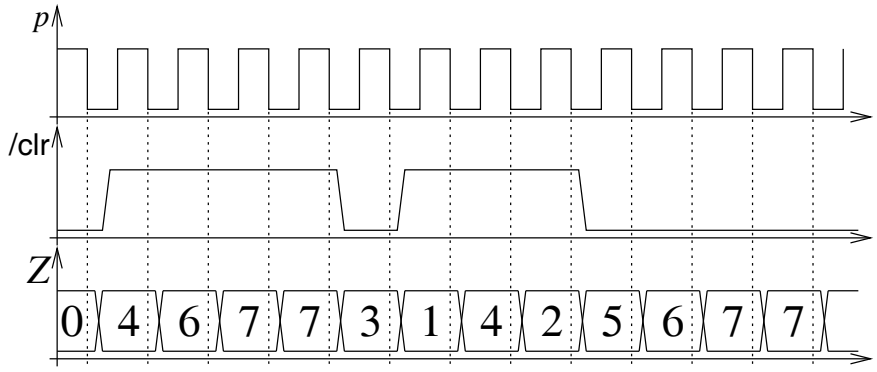

Figura 5: Temporizzazione di uscita.

25. Si veda la sezione 3.3 de "L'aritmetica dei calcolatori", Luigi Rizzo, disponibile sia sul sito che in biblioteca.

26. Per prima cosa scriviamo il numero 121 in binario. Si ha che $121_{10} = 1111001_2 = 1.111001_2 \cdot 2^6$. Quindi:

    - il bit di segno vale 0 perché il numero è positivo;
    - i bit di mantissa, che secondo lo standard sono 23, valgono $111001\underbrace{00000000000000000}_{17}$;
    - l'esponente è 6. Il fattore di polarizzazione (bias) è sempre $2^{K-1} - 1$. In questo caso il numero K di bit dell'esponente è 8, quindi il fattore di polarizzazione è 127. Il numero da rappresentare quindi è $6 + 127 = 133$ che, su 8 bit, si rappresenta come 10000101_2.

    La rappresentazione finale quindi è:

    $$0\,\underbrace{100\ 0010\ 1}_{\text{esponente}}\,\underbrace{111\ 0010\ 0000\ 0000\ 0000\ 0000}_{\text{mantissa}}$$

    e, raggruppando a gruppi di 4 bit, otteniamo la sua rappresentazione in esadecimale che è `0x42F20000`.

27. Osserviamo che il numero è negativo quindi il bit di segno sarà certamente 1. Il numero 1/16+1/4 è uguale a $2^{-2}+2^{-4}$ e quindi può essere scritto in binario nel seguente modo: $0.0101_2 = 1.01 \cdot 2^{-2}$. Alla luce di questo:

    - i bit di mantissa, che secondo lo standard sono 23, valgono $01\underbrace{00000000000000000000}_{21}$;
    - l'esponente è -2. Il fattore di polarizzazione (bias) è sempre $2^{K-1} - 1$. In questo caso il numero K di bit dell'esponente è 8, quindi il fattore di polarizzazione è 127. Il numero da rappresentare quindi è $-2 + 127 = 125$ che, su 8 bit, si rappresenta come 01111101_2.

    La rappresentazione finale quindi è:

    $$1\,\underbrace{011\ 1110\ 1}_{\text{esponente}}\,\underbrace{010\ 0000\ 0000\ 0000\ 0000\ 0000}_{\text{mantissa}}$$

    e, raggruppando a gruppi di 4 bit, otteniamo la sua rappresentazione in esadecimale che è `0xBEA00000`.

28. Dato il numero intero positivo X, sia N il numero minimo di bit in cui riesco a rappresentarlo in base 2, e x_i le cifre della rappresentazione, ovvero:

$$X = \sum_{i=0}^{N-1} x_i \, 2^i$$

Siccome N è il numero minimo di bit della rappresentazione, allora per forza deve essere $x_{N-1} = 1$ (altrimenti potrei scartarlo e rappresentare X con $N-1$ bit). Scrivo quindi:

$$X = \sum_{i=0}^{N-1} x_i \, 2^i = 2^{N-1} \left(x_{N-1} + \sum_{i=0}^{N-2} x_i \, 2^{-N+i+1} \right) = 2^{N-1} \left(1 + \sum_{i=0}^{N-2} x_i \, 2^{-N+i+1} \right)$$

che è proprio una notazione con esponente e mantissa adatta alla rappresntazione in floating point. Il numero intero positivo X è quindi rappresentabile se:

- l'esponente $N-1$ è sufficientemente piccolo da poter essere rappresentato sui K bit a disposizione;
- la mantissa $x_{N-2}, x_{N-3}, \ldots, x_1, x_0$ è sufficientemente corta da poter essere rappresentata con gli M bit a disposizione.

Non è invece rappresentabile se una delle due affermazioni precedenti è falsa.

Per quanto riguarda la prima condizione, il massimo esponente rappresentabile su K bit è $2^{K-1} - 1$. La condizione è violata quando:

$$\begin{aligned} N - 1 &> 2^{K-1} - 1 \\ N &> 2^{K-1} \\ N &\geq 2^{K-1} + 1 \end{aligned}$$

perché N è intero. Il numero minimo si ha quando tutti i bit della mantissa sono 0 e quindi il numero minimo in questa caso è:

$$2^{(2^{K-1}+1)-1} = 2^{2^{K-1}}$$

La seconda condizione diventa falsa non appena gli M bit della mantissa non sono più sufficienti a contenere i bit x_i. Siccome i bit x_i sono $N-1$, il numero X non è più rappresentabile per indufficienza di bit nella mantissa quando:

$$\begin{aligned} N - 1 &> M \\ N &> M + 1 \\ N &\geq M + 2 \end{aligned}$$

Fra tutti i numeri che verificano la condizione precedente, il minimo capita quando tutti i bit x_i sono 0 tranne il meno significativo che deve valere 1. Questo vuol dire $x_0 = 1$, $x_i = 1$ per $i \neq 0$. Tale numero vale:

$$2^{(M+2)-1} \left(1 + \sum_{i=0}^{(M+2)-2} x_i \, 2^{-(M+2)+i+1} \right) = 2^{M+1} \left(1 + 2^{-M-1} \right) = 2^{M+1} + 1$$

Siccome entrambi sono numeri non rappresentabili allora il minimo è dato da $\min\{2^{2^{K-1}}, 2^{M+1} + 1\}$. Nel caso dei `float`, in cui $K = 8$ e $M = 23$, questo numero vale $2^{24}+1$. Nel caso dei `double`, in cui $K = 11$ e $M = 52$, questo numero vale $2^{53} + 1$.

Stato	y_2	y_1	y_0	carattere	Z
S_0	0	0	0	C	0x43
S_1	0	0	1	i	0x69
S_2	0	1	0	a	0x61
S_3	0	1	1	o	0x6F
S_4	1	0	0	Space	0x20
S_5	1	0	1	Z	0x5A
S_6	1	1	0	i	0x69
S_7	1	1	1	o	0x6F

Tabella 1: Codifica degli stati e evoluzione delle uscite.

29. Per prima cosa si osserva che la stringa è composta da otto caratteri (si deve includere anche lo spazio intermedio).

    La legge di transizione degli stati che andremo a realizzare è esattamente uguale a quella di un contatore con segnale /init. Difatti una volta emesso un carattere si passa al seguente e così via. Una codifica intuitiva è quella di Tabella 1. La legge di evoluzione delle uscite può quindi essere implementata da una ROM dove si usa la codifica dello stato come indirizzo e la codifica ASCII come dato da scrivere nel relativo indirizzo. Lo schema è riportato in Figura 6.

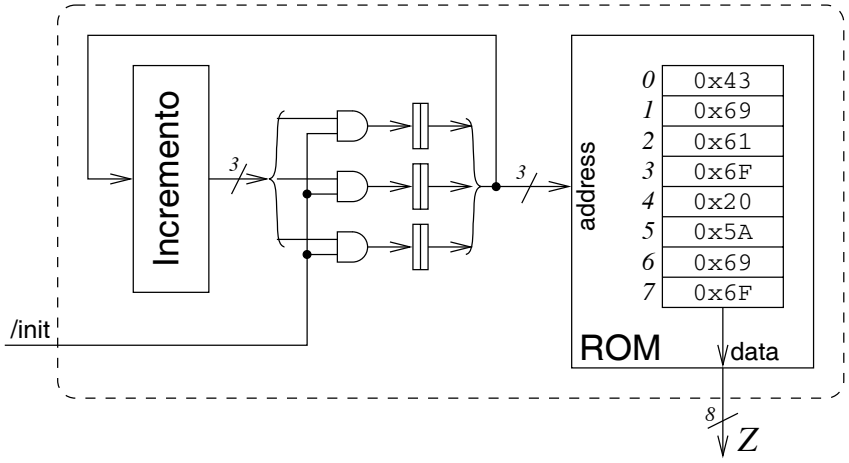

Figura 6: Schema del produttore di stringa.

30. Per quanto riguarda la parte intera del numero si può dire che: $125_{10} = 111\ 1101_2 = \texttt{7D}_{16}$. Siccome abbiamo a disposizione 16 bit la parte intera in esadecimale è $\texttt{007D}_{16}$.

    Per quanto riguarda la parte frazionaria $0.125_{10} = 0.001_2$. Siccome abbiamo a disposizione 16 bit, la parte frazionaria in esadecimale è $\texttt{2000}_{16}$. In fixed point un numero si rappresenta accostando parte intera e parte frazionaria, quindi la rappresentazione del numero è $\texttt{007D2000}_{16}$ che occupa 4 byte.

    Se inizio a scrivere il numero dalla locazione 0x0C004C08 mi servono quattro locazioni di memoria a partire da questo indirizzo, ovvero: 0x0C004C08, 0x0C004C09, 0x0C004C0A e 0x0C004C0B.

    Se l'ordinamento dei byte è big endian allora:

    - 00 → 0x0C004C08,
    - 7D → 0x0C004C09,
    - 20 → 0x0C004C0A,
    - 00 → 0x0C004C0B.

Se l'ordinamento dei byte è little endian allora:

- `00` → `0x0C004C08`,
- `20` → `0x0C004C09`,
- `7D` → `0x0C004C0A`,
- `00` → `0x0C004C0B`.

31. Dalla soluzione dell'esercizio precedente segue che $125.125_{10} = 1111101.001_2$. Se mettiamo in evidenza la massima potenza di 2 possibile otteniamo $1.111101001 \cdot 2^6$. Quindi:

    - il numero è positivo quindi il bit di segno vale 0;
    - l'esponente, secondo lo standard dei `float`, si rappresenta su $K = 8$ bit, con fattore di polarizzazione $2^{K-1} - 1 = 127$. Quindi il numero da rappresentare sugli 8 bit di esponente è $6 + 127 = 133 = 10000101_2$;
    - la mantissa è su 23 bit. Quindi usiamo quelli che abbiamo a disposizione e riempiamo quelli meno significativi con 0. Ovvero si ottiene $\underbrace{111101001\overbrace{00000000000000}^{14}}_{23}$.

    Se lo scrivo tutto insieme ottengo:

    $$0\,\underbrace{100\ 0010\ 1}_{\text{esponente}}\underbrace{111\ 1010\ 0100\ 0000\ 0000\ 0000}_{\text{mantissa}}$$

    che in esadecimale si scrive `0x42FA4000`. Il numero occupa 4 byte e tante sono le locazioni di memoria di cui ho bisogno per memorizzarlo. Se l'ordinamento dei byte è big endian allora:

    - `42` → `0x0C004C08`,
    - `FA` → `0x0C004C09`,
    - `40` → `0x0C004C0A`,
    - `00` → `0x0C004C0B`.

    Se l'ordinamento dei byte è little endian allora:

    - `00` → `0x0C004C08`,
    - `40` → `0x0C004C09`,
    - `FA` → `0x0C004C0A`,
    - `42` → `0x0C004C0B`.

32. Siano $X^S = x_{31}$, $X^E = (x_{30}, \ldots, x_{23})$, $X^F = (x_{22}, \ldots, x_0)$, ovvero X_S è il bit di segno di X, X^E sono i bit di esponente di X e X^F sono i bit di mantissa di X. Equivalentemente, siano $Z^S = z_{63}$ il bit di segno di Z, $Z^E = (z_{62}, \ldots, z_{52})$ i bit di esponente di Z e $Z^F = (z_{51}, \ldots, z_0)$ i bit della parte frazionaria di Z. L'obiettivo dell'esercizio è produrre Z, ovvero gli sua componente (Z^S, Z^E, Z^F), a partire da X, tale che Z rappresenti in `double` lo stesso valore che X rappresenta in `float`.

    Siccome il segno deve essere lo stesso allora $Z^S = X^S$ e quindi $z_{63} = x_{31}$.

    Per l'esponente è richiesta un po' di cautela. Alla luce dello standard IEEE 754, sappiamo che il valore E dell'esponente è rappresentato da:

    $$X^E = E + (2^{K-1} - 1) = E + (2^7 - 1) \tag{1}$$

    dove $K = 8$ è il numero di bit usati per rappresentare l'esponente. Quindi, partendo dalla sua rappresentazione, l'esponente E è uguale a:

    $$E = X^E - (2^7 - 1)$$

Siccome X e Z devono rappresentare lo stesso numero decimale, devono avere anche uguale esponente. Quindi:

$$Z^E = E + (2^{10} - 1) = X^E - (2^7 - 1) + (2^{10} - 1) = X^E - 2^7 + 1 + 2^{10} - 1 = \\ X^E - 2^7 + 2^{10} = X^E + 2^7(2^3 - 1)$$

Scrivendo X^E con tutti i suoi bit si ricava che:

$$Z^E = X^E + 2^7(2^3 - 1) = \sum_{i=0}^{7} x_{23+i}\, 2^i + 2^7(2^3 - 1) = \\ \sum_{i=0}^{6} x_{23+i}\, 2^i + x_{30}\, 2^7 + 2^7(2^3 - 1) = \\ \sum_{i=0}^{6} x_{23+i}\, 2^i + 2^7(2^3 - 1 + x_{30}) =$$

Dall'ultima espressione si ricavano certamente i 7 bit meno significativi di Z^E, che coincidono con i 7 bit meno significativi di X^E, ovvero:

$$z_{52+i} = x_{23+i} \qquad i = 0, 1, 2, 3, 4, 5, 6$$

Gli altri 4 bit più significativi di Z^E, che sono $(z_{62}, z_{61}, z_{60}, z_{59})$, sono dati dall'espressione:

$$2^3 - 1 + x_{30}$$

Quando $x_{30} = 0$ allora valgono $2^3 - 1 = 0111_2$. Quando $x_{30} = 1$ allora valgono $2^3 = 1000_2$. Si possono esprimere entrambe le configurazioni scrivendole come funzione di x_{30} nel seguente modo:

$$z_{62} = x_{30} \\ z_{59+i} = \overline{x_{30}} \qquad i = 0, 1, 2$$

Infine per quello che riguarda la mantissa la soluzione è semplice: si tratta di ricopiare i bit di mantissa di X^F sui bit più significativi di Z^F e riempire i bit meno significativi con bit 0. Quindi:

$$z_{29+i} = x_i \quad i = 0, 1, 2, \ldots, 21, 22 \\ z_i = 0 \qquad i = 0, 1, 2, \ldots, 27, 28$$

che risolve il problema.

Esercizi non risolti

1. È noto che i codici operativi delle istruzioni in e out, possono essere: 0xE4, 0xE5, 0xE6, 0xE7, 0xEC, 0xED, 0xEE, 0xEF. Realizzare la rete logica che, dati in ingresso gli 8 bit di un codice operativo restituisce 1 se l'operazione è di in oppure out, 0 altrimenti.

2. In Tabella 2 si riportano alcune istruzioni presenti in memoria: nella terza colonna è riportata l'istruzione vera e propria, nella seconda la sua codifica e infine la prima colonna contiene l'indirizzo di memoria dove l'istruzione è memorizzata.

Indirizzo Mem. (Hex)	Codifica (Hex)	Istruzione
0x00000062	8B 18	movl (%eax), %ebx
0x00000064	BB 04 03 02 01	movl $0x01020304, %ebx
0x00000069	29 C3	subl %eax, %ebx

Tabella 2: Codice in memoria.

Appena prima della fase di fetch della prima istruzione del codice, il processore si trova nel seguente stato:

- il registro EIP, instruction pointer, contiene il valore 0x00000062 (che è il valore corretto affinché possa avvenire il fetch all'indirizzo della prima istruzione);
- il registro EAX contiene il valore 0x00000062.

Quale è il contenuto dei registri EIP, EAX e EBX, dopo l'esecuzione del codice?

3. **Difficile.** In Tabella 3 si riportano alcune istruzioni presenti in memoria: nella terza colonna è riportata l'istruzione vera e propria, nella seconda la sua codifica e infine la prima colonna contiene l'indirizzo di memoria dove l'istruzione è memorizzata.

Indirizzo Mem. (Hex)	Codifica (Hex)	Istruzione
0x00000058	00 C4	addb %al, %ah
0x0000005A	8B 18	movl (%eax), %ebx
0x0000005C	81 D3 08 04 02 01	adcl $0x01020408, %ebx

Tabella 3: Codice in memoria.

Appena prima della fase di fetch della prima istruzione del codice, il processore si trova nel seguente stato:

- il registro EIP, instruction pointer, contiene il valore 0x00000058 (che è il valore corretto affinché possa fare il fetch all'indirizzo della prima istruzione);
- il registro EAX contiene il valore 0x0000A25E.

Quale è il contenuto dei registri EIP, EAX e EBX, dopo l'esecuzione del codice?

4. Riempire opportunamente il codice riportato sotto, dove appare il carattere "_". Riportare, ove richiesto, o i codici operativi mancanti o l'istruzione. Si tenga conto che sulla sinistra sono elencati i codici operativi e sulla destra le istruzioni relative.

```
B0 24                   movb    $0x24,  %al
B0 25                   movb    $0x25,  %al
__ AC                   movb    $0xAC,  %al
B0 __                   movb    $0x02,  %al
B0 __                   movb    $24,    %al
__ __                   movb    $128,   %al
```

```
B4 AC                movb    $0xAC,  %ah
B3 AC                movb    $0xAC,  %bl
B7 AC                movb    $0xAC,  %bh
B1 AC                movb    $0xAC,  %cl
B5 AC                movb    $0xAC,  %ch
B2 AC                movb    $0xAC,  %dl
B6 AC                movb    $0xAC,  %dh
88 E0                movb    %ah,    %al
88 D8                movb    %bl,    %al
88 F8                movb    %bh,    %al
88 C8                movb    %cl,    %al
88 __                movb    %ch,    %al
88 C4                movb    %al,    %ah
88 C3                movb    %al,    %bl
88 C7                movb    %al,    %bh
88 C1                movb    %al,    %cl
88 C5                movb    %al,    %ch
88 C0                ------------------
04 24                addb    $0x24,  %al
04 25                addb    $0x25,  %al
__ AC                addb    $0xAC,  %al
04 02                addb    $0x02,  %al
04 __                addb    $24,    %al
04 __                addb    $128,   %al
00 E0                addb    %ah,    %al
00 D8                addb    %bl,    %al
00 F8                addb    %bh,    %al
00 C8                addb    %cl,    %al
__ __                addb    %ch,    %al
00 C4                addb    %al,    %ah
00 C3                addb    %al,    %bl
00 C7                addb    %al,    %bh
00 C1                addb    %al,    %cl
00 __                addb    %al,    %ch
```

5. Abbiamo a disposizione un processore con quattro sole istruzioni: `store`, `load`, `get`, `put`. La tabella 4 riporta i loro codici operativi e la lunghezza totale delle istruzioni in bit.

Cod. operativo	Istruzione	Lunghezza (bit)
00	store	5
01	load	2
10	put	4
11	get	3

Tabella 4: Codifiche delle istruzioni.

Per il prelievo delle istruzioni (fase di fetch), esiste un dispositivo che preleva i primi due bit dell'istruzione (codice operativo) e, in base ad essi, produce il numero di ulteriori bit da prelevare. Realizare tale dispositivo.

Compitino di Calcolatori Elettronici del 05/11/2004

Preambolo Sia C l'ultima cifra del tuo numero di matricola, sia $V = C+5$ e siano (v_3, v_2, v_1, v_0) i bit della sua rappresentazione con v_3 bit **più** significativo e v_0 il **meno** significativo. Scriverli sotto perché nel compitino ne sarà richiesto l'utilizzo:

$V = C + 5 =$_____, rappresentato da $v_3 =$___, $v_2 =$___, $v_1 =$___, $v_0 =$___.

Esercizio 1 Si vuole realizzare la rete combinatoria che, dato un numero X rappresentato su 4 bit, ritorna un segnale z che è: 1 se $X \in [V-4, V]$ (per il tuo valore di V si veda il Preambolo), 0 altrimenti. Per esempio se il tuo V vale 14 allora la rete ritorna 1 se $X \in \{10, 11, 12, 13, 14\}$ e ritorna 0 se $X \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 15\}$.

Scrivere z in forma minima SP (somma di prodotti) e PS (prodotti di somme).

Esercizio 2 Ipotiziamo che le porte logiche producano il risultato in uscita dopo un tempo Δ dal momento in cui gli ingressi sono validi, e che il valore dell'uscita sia **instabile** dal momento in cui un ingresso cambia al momento in cui l'uscita assume il nuovo valore (si veda la Figura 7 per la temporizzazione di una porta XOR di questo tipo).

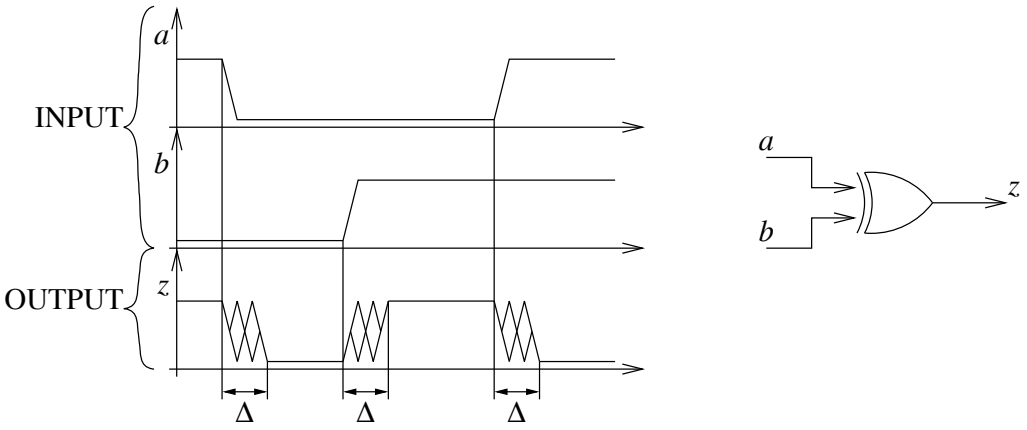

Figura 7: Temporizzazione della porta XOR.

Abbiamo realizzato il circuito di Figura 8 (dove v_1 e v_0 sono i valori di bit trovato nel Preambolo). Disegnare le temporizzazioni dei segnali x, y e z, in corrispondenza degli ingressi riportati nella figura stessa, fornendo adeguata motivazione.

Esercizio 3 Si realizzi in contatore su due bit che conta **a cominciare dal numero** 3. Questo vuole dire che il contatore inizia con 3 e poi continua con 0, 1, 2, poi di nuovo 3 e così via. Il contatore è inoltre dotato di un segnale di ingresso init (attivo alto) che, quando marcato su valore 1, imposta il valore iniziale del contatore che, in questo caso, è 3.

Per la realizzazione del contatore si possono utilizzare soltanto: le porte logiche AND, OR, NOT, NAND, NOR e XOR e registri ad un bit.

Sol. Compitino di Calcolatori Elettronici del 05/11/2004

Esercizio 1 Si riportano in Tabella 5 le varie tabelle di verità relative ai differenti esercizi per i differenti valori di V. z_V rappresenta l'uscita per lo specifico valore di V.

Si risolve applicando le mappe di Karnaugh. Si trova che:

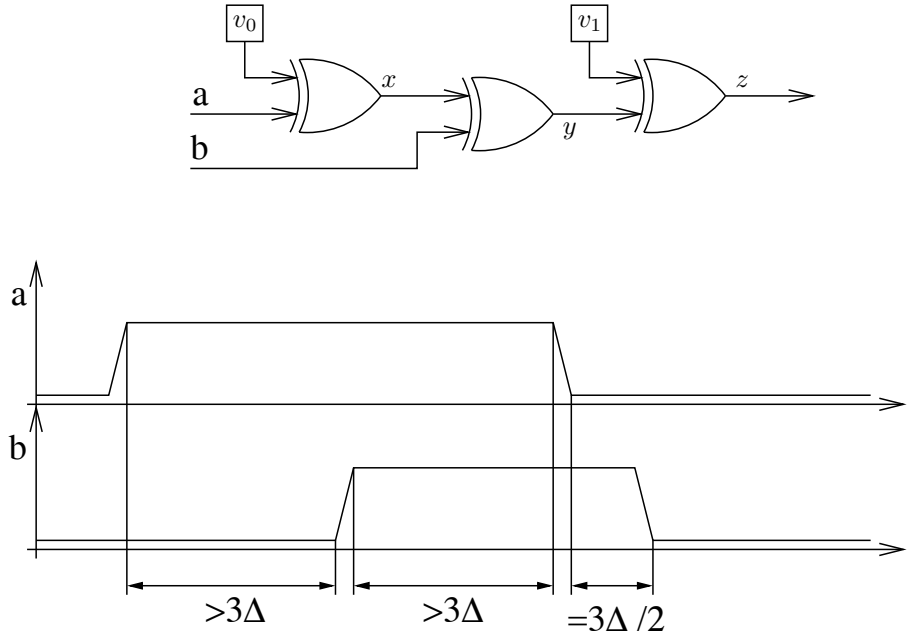

Figura 8: Schema Esercizio 2.

x_3	x_2	x_1	x_0	z_5	z_6	z_7	z_8	z_9	z_{10}	z_{11}	z_{12}	z_{13}	z_{14}
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	0	0	0	0	0	0
0	1	0	1	1	1	1	1	1	0	0	0	0	0
0	1	1	0	0	1	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	1	1	1	0	0	0
1	0	0	0	0	0	0	1	1	1	1	1	0	0
1	0	0	1	0	0	0	0	1	1	1	1	1	0
1	0	1	0	0	0	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	1	1	1
1	1	0	1	0	0	0	0	0	0	0	0	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	0	0	0	0

Tabella 5: Tabella di verità, Es. 1.

- Forma SP $z_5 = \overline{x_3}\, x_2\, \overline{x_1} + \overline{x_3}\, \overline{x_1}\, x_0 + \overline{x_3}\, \overline{x_2}\, x_1$ oppure, $z_5 = \overline{x_3}\, x_2\, \overline{x_1} + \overline{x_3}\, \overline{x_2}\, x_0 + \overline{x_3}\, \overline{x_2}\, x_1$. Forma PS $z_5 = \overline{x_3}(\overline{x_2} + \overline{x_1})(x_2 + x_1 + x_0)$.
- Forma SP $z_6 = \overline{x_3}\, x_2\, \overline{x_1} + \overline{x_3}\, x_1\, \overline{x_0} + \overline{x_3}\, \overline{x_2}\, x_1$, oppure $z_6 = \overline{x_3}\, x_2\, \overline{x_0} + \overline{x_3}\, x_2\, \overline{x_1} + \overline{x_3}\, \overline{x_2}\, x_1$. Forma PS $z_6 = \overline{x_3}(x_2 + x_1)(\overline{x_2} + \overline{x_1} + \overline{x_0})$.
- Forma SP $z_7 = \overline{x_3}\, x_2 \; + \overline{x_3}\, x_1\, x_0$. Forma PS $z_7 = \overline{x_3}(x_2 + x_1)(x_2 + x_0)$;
- Forma SP $z_8 = \overline{x_3}\, x_2 + x_3\, \overline{x_2}\, \overline{x_1}\, \overline{x_0}$. Forma PS $z_8 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_3} + \overline{x_0})(\overline{x_3} + \overline{x_1})$, oppure $z_8 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_2 + \overline{x_0})(\overline{x_3} + \overline{x_1})$, oppure $z_8 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_2 + \overline{x_0})(x_2 + \overline{x_1})$, oppure $z_8 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_3} + \overline{x_0})(x_2 + \overline{x_1})$.

- Forma SP $z_9 = x_3\,\overline{x_2}\,\overline{x_1} + \overline{x_3}\,x_2\,x_0 + \overline{x_3}\,x_2\,x_1$. Forma PS $z_9 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_3 + x_1 + x_0)(\overline{x_3} + \overline{x_1})$, oppure $z_9 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_2} + x_1 + x_0)(\overline{x_3} + \overline{x_1})$, oppure $z_9 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_3 + x_1 + x_0)(x_2 + \overline{x_1})$, oppure $z_9 = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_2} + x_1 + x_0)(x_2 + \overline{x_1})$.

- Forma SP $z_{10} = x_3\,\overline{x_2}\,\overline{x_1} + \overline{x_3}\,x_2\,x_1 + x_3\,\overline{x_2}\,\overline{x_0}$. Forma PS $z_{10} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_3 + x_1)(\overline{x_3} + \overline{x_1} + \overline{x_0})$, oppure $z_{10} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_2} + x_1)(\overline{x_3} + \overline{x_1} + \overline{x_0})$, oppure $z_{10} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_3 + x_1)(x_2 + \overline{x_1} + \overline{x_0})$, oppure $z_{10} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_2} + x_1)(x_2 + \overline{x_1} + \overline{x_0})$.

- Forma SP $z_{11} = x_3\,\overline{x_2} + \overline{x_3}\,x_2\,x_1\,x_0$. Forma PS $z_{11} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_3 + x_1)(x_3 + x_0)$, oppure $z_{11} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_2} + x_1)(x_3 + x_0)$, oppure $z_{11} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(x_3 + x_1)(\overline{x_2} + x_0)$, oppure $z_{11} = (x_3 + x_2)(\overline{x_3} + \overline{x_2})(\overline{x_2} + x_1)(\overline{x_2} + x_0)$.

- Forma SP $z_{12} = x_3\,\overline{x_2} + x_3\,\overline{x_1}\,\overline{x_0}$. Forma PS $z_{12} = x_3(\overline{x_2} + \overline{x_0})(\overline{x_2} + \overline{x_1})$.

- Forma SP $z_{13} = x_3\,x_2\,\overline{x_1} + x_3\,\overline{x_2}\,x_1 + x_3\,\overline{x_1}\,x_0$, oppure $z_{13} = x_3\,x_2\,\overline{x_1} + x_3\,\overline{x_2}\,x_1 + x_3\,\overline{x_2}\,x_0$. Forma PS $z_{13} = x_3(\overline{x_2} + \overline{x_1})(x_2 + x_1 + x_0)$.

- Forma SP $z_{14} = x_3\,x_2\,\overline{x_1} + x_3\,\overline{x_2}\,x_1 + x_3\,x_1\,\overline{x_0}$, oppure $z_{14} = x_3\,x_2\,\overline{x_1} + x_3\,\overline{x_2}\,x_1 + x_3\,x_2\,\overline{x_0}$. Forma PS $z_{14} = x_3(\overline{x_2} + \overline{x_1} + \overline{x_0})(x_2 + x_1)$.

Esercizio 2 La temporizzazione è illustrata in Figura 9, dove i valori binari sono espressi mediante funzioni booleane di v_1 e v_0 che variano da compito a compito.

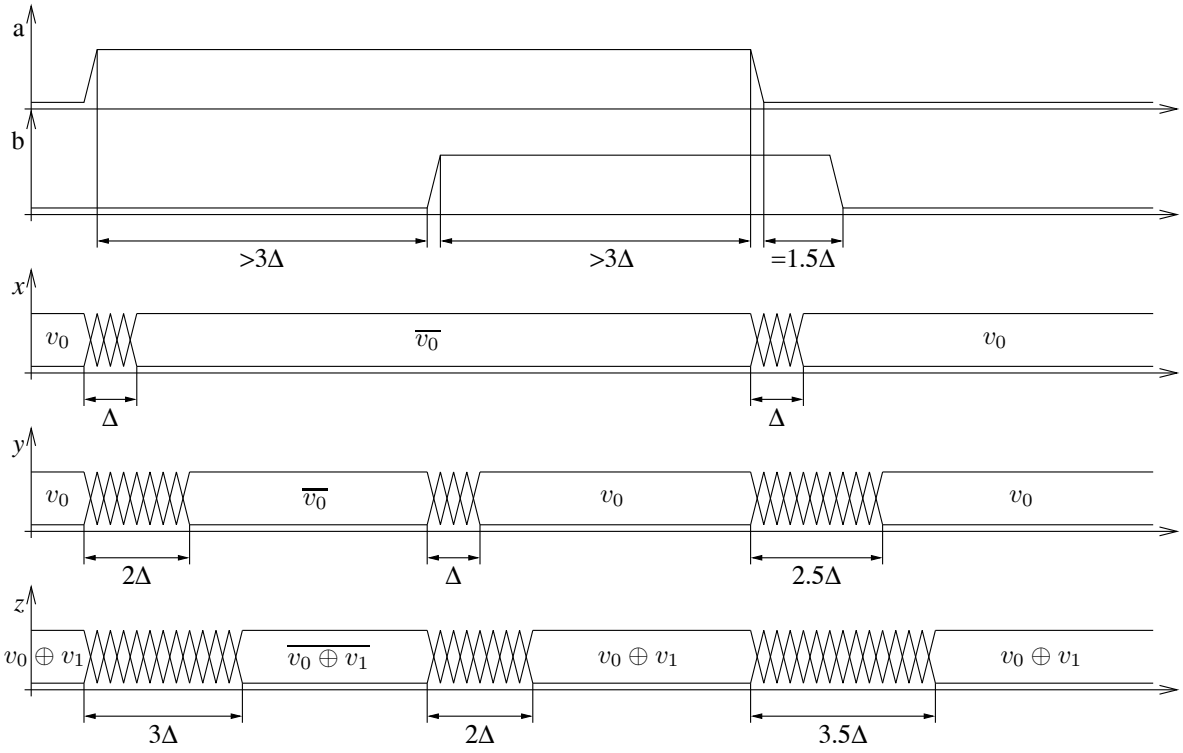

Figura 9: Temporizzazione dell'Esercizio 2.

Si osserva che:

- l'instabilità di z dovuta alla prima transizione di a è lunga 3Δ perché si devono attraversare 3 livelli di logica.;

- l'instabilità di z dovuta alla prima transizione di b è lunga 2Δ perché si devono attraversare 3 livelli di logica.;

- la durata della terza instabilità è 3.5Δ perché, dopo la transizione $1 \rightarrow 0$ di a, ancor prima che y possa assumere il nuovo valore anche b cambia valore. In questo caso quindi si dovrà attendere Δ dall'ultima transizione (quella di b) e quindi 2.5Δ dopo la transizione di a. Conseguentemente z assume il nuovo valore dopo 3.5Δ;
- i valori di y e di z prima e dopo l'ultima transizione sono **uguali** (e non opposti come nelle prime due transizioni). Il valore prima infatti è $(((v_0 \oplus 1) \oplus 1) \oplus v_1 = v_0 \oplus v_1$. Il valore dopo è $(((v_0 \oplus 0) \oplus 0) \oplus v_1 = v_0 \oplus v_1$.

Esercizio 3 Il dispositivo è dotato del segnale init di ingresso, e del segnale p di marcatura. Ha inoltre due bit in uscita (z_1, z_0) che rappresentano il valore corrente Z del contatore.

Per prima cosa si osserva che, quando il segnale init non è attivo (ovvero vale 0) il dispositivo si comporta esattamente come un normale contatore a 2 bit, ovvero il prossimo valore sarà quello corrente incrementato di 1. La tabella di verità del circuito di incremento è riportata in Tabella 6.

x_1	x_0	z_1	z_0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Tabella 6: Tabella di verità dell'incremento.

Dalla tabella di verità segue immediatamente che: $z_0 = \overline{x_0}$ e $z_1 = \overline{x_1}\, x_0 + x_1\, \overline{x_0} = x_1 \oplus x_0$.

Quando invece il segnale init vale 1, il valore delle uscite viene impostato a 3. Questo suggerisce che init deve essere collegato in OR alle uscite del ciruito di incremento, in modo tale che quando vale 1, il prossimo valore dell'uscita sarà comunque 3.

Da questa descrizione si può ricavare direttamente la circuiteria, riportata in Figura 10.

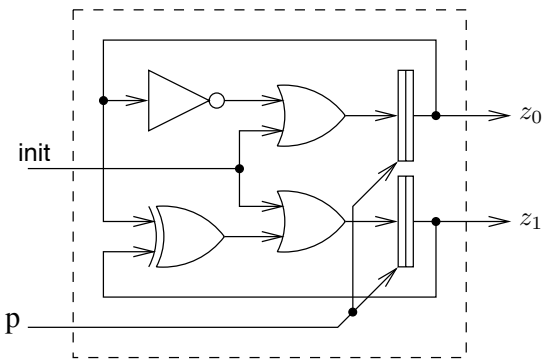

Figura 10: Schema del dispositivo dell'Esercizio 3.

Compitino di Calcolatori Elettronici del 25/11/2004

Preambolo Sia C la **penultima** cifra del tuo numero di matricola, sia $V = C + 5$ e siano (v_3, v_2, v_1, v_0) i bit della sua rappresentazione con v_3 bit **più** significativo e v_0 il **meno** significativo. Scriverli sotto perché nel compitino ne sarà richiesto l'utilizzo:

$V = C + 5 =$_____, rappresentato da $v_3 =$___, $v_2 =$___, $v_1 =$___, $v_0 =$___.

Esercizio 1 Sia X un numero intero di 4 bit rappresentato in complemento a 2 (quindi da -8 a 7). Si realizzi la rete logica che dato in ingresso X produce:

- il segnale di un bit v che vale 1 quando X è rappresentabile anche su 3 bit, 0 quando non è rappresentabile;
- il numero Z rappresentato in complemento a 2 su 3 bit che, nel caso in cui X sia rappresentabile su 3 bit, ritorna tale valore. Nel caso in cui X non sia rappresentabile su 3 bit, il valore di Z è indifferente.

Esercizio 2 È dato il circuito di Figura 11, dotato di un segnale di controllo ctrl ad un bit in ingresso. Il circuito produce un segnale Z su 4 bit in uscita. Il circuito è inoltre dotato del segnale di

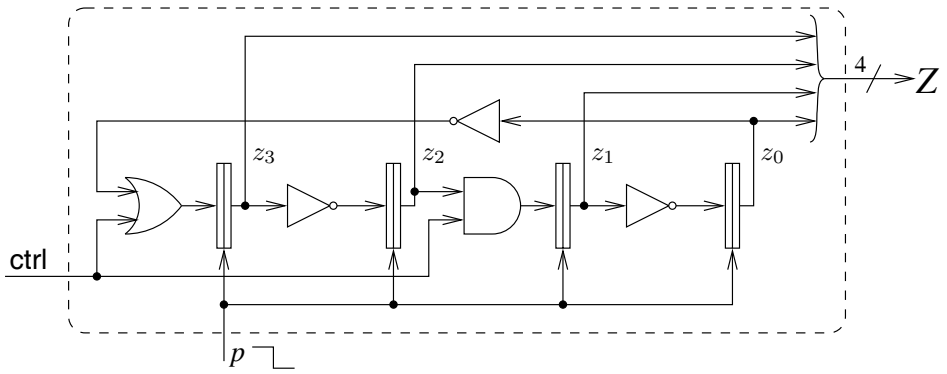

Figura 11: Schema Esercizio 2.

marcatura p che marca lo stato dei registri sui fronti di discesa (da 1 a 0).

All'istante iniziale, i valori dei registri ad un bit sono $z_3 = v_3$, $z_2 = v_2$, $z_1 = v_1$ e $z_0 = v_0$, di modo che il valore dell'uscita Z è V (si veda il Preambolo per il propri valori di V e v_i). All'istante iniziale il segnale di controllo ctrl vale 0.

Si determini come si evolvono i valori di z_3, z_2, z_1 e z_0, specificando il valore numerico di Z, in presenza degli ingressi di Figura 12.

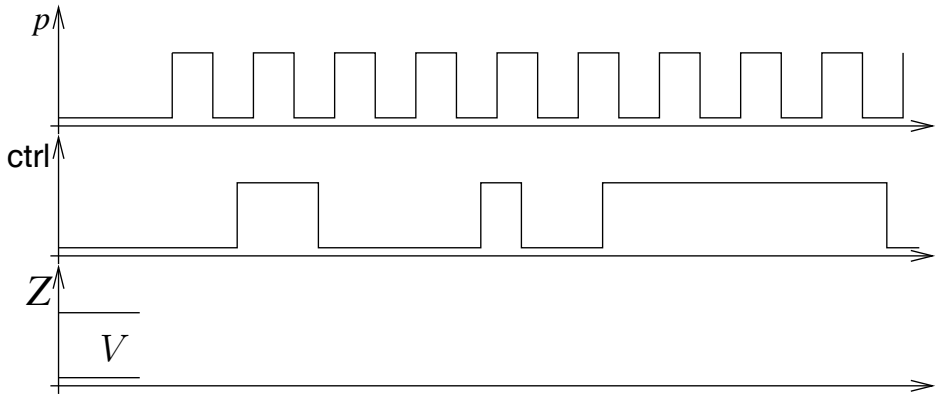

Figura 12: Temporizzazione dei segnali di ingresso. Esercizio 2.

Esercizio 3 In Tabella 7 si riportano alcune istruzioni presenti in memoria: nella terza colonna è riportata l'istruzione vera e propria, nella seconda la sua codifica e infine la prima colonna contiene l'indirizzo di memoria dove l'istruzione è memorizzata.

Ind. Memoria (Hex)	Codifica (Hex)	Istruzione
0x00000009	B9 80 FF FF FF	movl $-128, %ecx
0x0000000E	FE C1	incb %cl
0x00000010	29 C0	subl %eax, %eax
0x00000012	A0 0B 00 00 00	movb 11, %al
0x00000017	01 C8	addl %ecx, %eax

Tabella 7: Codice in memoria. Esercizio 3.

Appena prima della fase di fetch della prima istruzione del codice, il processore si trova nel seguente stato:

- il registro `EIP`, instruction pointer, contiene il valore `0x00000009` (che è il valore corretto affinché possa avvenire il fetch all'indirizzo della prima istruzione);
- il registro `EAX` contiene il valore `0x00000062`.

Determinare il contenuto dei registri `EIP`, `EAX`, `ECX` al termine di **ogni istruzione**.

Sol. Compitino di Calcolatori Elettronici del 25/11/2004

Esercizio 1 La rete da realizzare è combinatoria. La tabella di verità è riportata in Tabella 8.

X	x_3	x_2	x_1	x_0	v	Z	z_2	z_1	z_0
-8	1	0	0	0	0	–	–	–	–
-7	1	0	0	1	0	–	–	–	–
-6	1	0	1	0	0	–	–	–	–
-5	1	0	1	1	0	–	–	–	–
-4	1	1	0	0	1	-4	1	0	0
-3	1	1	0	1	1	-3	1	0	1
-2	1	1	1	0	1	-2	1	1	0
-1	1	1	1	1	1	-1	1	1	1
0	0	0	0	0	1	0	0	0	0
1	0	0	0	1	1	1	0	0	1
2	0	0	1	0	1	2	0	1	0
3	0	0	1	1	1	3	0	1	1
4	0	1	0	0	0	–	–	–	–
5	0	1	0	1	0	–	–	–	–
6	0	1	1	0	0	–	–	–	–
7	0	1	1	1	0	–	–	–	–

Tabella 8: Tabella di verità. Esercizio 1.

Ispezionando semplicemente la tabella si ricava che:

- $v = x_3\, x_2 + \overline{x_3}\,\overline{x_2} = \overline{x_3 \oplus x_2}$;
- $z_2 = x_2$;
- $z_1 = x_1$;
- $z_0 = x_0$.

Esercizio 2 Per prima cosa si rammenta che il segnale p marca sui **fronti di discesa** da 1 a 0, quindi i valori marcati di ctrl sono nell'ordine: $0, 1, 0, 0, 0, 1, 1, 1, 1$ (se, erroneamente avessimo marcato sui fronti di salita avremmo ottenuto $0, 1, 0, 0, 1, 0, 1, 1, 1$).

L'evoluzione temporale dei segnali z_i è quindi quella riportata in Tabella 9, ove la prima corrisponde allo stato iniziale.

					Valore di Z per ogni V									
z_3	z_2	z_1	z_0	crtl	5	6	7	8	9	10	11	12	13	14
v_3	v_2	v_1	v_0	0	5	6	7	8	9	10	11	12	13	14
$\overline{v_0}$	$\overline{v_3}$	0	$\overline{v_1}$	1	5	12	4	9	1	8	0	9	1	8
1	v_0	$\overline{v_3}$	1	0	15	11	15	9	13	9	13	9	13	9
0	0	0	v_3	0	0	0	0	1	1	1	1	1	1	1
$\overline{v_3}$	1	0	1	0	13	13	13	5	5	5	5	5	5	5
0	v_3	0	1	1	1	1	1	5	5	5	5	5	5	5
1	1	v_3	1	1	13	13	13	15	15	15	15	15	15	15
1	0	1	$\overline{v_3}$	1	11	11	11	10	10	10	10	10	10	10
1	0	0	0	1	8	8	8	8	8	8	8	8	8	8

Tabella 9: Andamento delle uscite. Esercizio 2.

Esercizio 3 I valori dei registri sono quelli riportati in Tabella 10.

Stato	EIP (Hex)	EAX (Hex)	ECX (Hex)
Inizio	00 00 00 09	00 00 00 62	unknown
dopo 1^a istr.	00 00 00 0E	00 00 00 62	FF FF FF 80
dopo 2^a istr.	00 00 00 10	00 00 00 62	FF FF FF 81
dopo 3^a istr.	00 00 00 12	00 00 00 00	FF FF FF 81
dopo 4^a istr.	00 00 00 17	00 00 00 FF	FF FF FF 81
dopo 5^a istr.	00 00 00 19	00 00 00 80	FF FF FF 81

Tabella 10: Tracing dei registri. Esercizio 3.

Prova scritta di Calcolatori Elettronici del 15/12/2004

Esercizio 1 In memoria, a partire dall'indirizzo 0x0000C000, è contenuto un testo, codificato con codifica ASCII. Scrivere un programma in Assembly che, a partire da tale indirizzo, scorre il testo fino a quando non incontra il primo carattere "X" (il cui codice ASCII è 88 = 0x58). Dopo che ha incontrato questo carattere, il registro EBX contiene l'indirizzo di memoria dove è contenuto.

Esercizio 2 È noto che la maschera di un dispositivo di I/O è una rete che produce 0 in corrispondenza degli indirizzi delle porte del dispositivo stesso. Si realizzi una maschera che riconosce gli indirizzi delle porte dalla 0x0578 alla 0x057B (quindi la maschera deve riconoscere le quattro porte con indiririzzo: 0x0578, 0x0579, 0x057A e 0x057B).

Esercizio 3 Realizzare un contatore da 1 a 4, nel senso che il contatore produce $1, 2, 3, 4, 1, 2, \ldots$ e così via (notare che solitamente il contatore inizia a contare da 0). Il contatore è inoltre dotato di un segnale di /clr attivo basso che quando marcato su valore 0 riporta il contatore al valore 1.

Per la realizzazione del dispositivo di possono utilizzare soltanto porte logiche elementari (AND, OR, NOT), porte XOR e NXOR e registri.

Sol. Prova scritta di Calcolatori Elettronici del 15/12/2004

Esercizio 1 L'idea è molto semplice: scorrere il testo fino a quando non si incontra il carattere "X", poi uscire. Questo equivale al seguente codice:

```
movl    $0x0000BFFF, %ebx
incl    %ebx
cmpb    $88, (%ebx)
jnz     <istruzione incl>
```

Esercizio 2 Analizzando i valori degli indirizzi da riconoscere si ricava immediatamente l'uscita della maschera:

$$z = x_{15} + x_{14} + x_{13} + x_{12} + x_{11} + \overline{x_{10}} + x_9 + \overline{x_8} + x_7 + \overline{x_6} + \overline{x_5} + \overline{x_4} + \overline{x_3} + x_2$$

Esercizio 3 Si può risolvere codificando su due o tre bit. Su tre bit non c'è la rete di uscita. Su due bit la soluzione è più compatta. Si riporta la soluzione dove si assume la codifica: 00 per lo stato in cui esce 1, 01 per lo stato in cui esce 2, 10 per lo stato in cui esce 3 e 11 per lo stato in cui esce 4.

Adottando questa soluzione le transizioni sono quelle classiche del contatore. La differenza è che l'uscita deve passare da un'ulteriore rete che incrementa la codifica dello stato di uno. Quindi si ottiene la rete sequenziale di Figura 13.

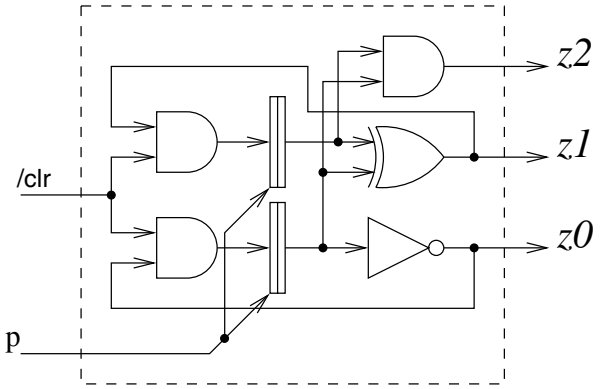

Figura 13: Soluzione dell'esercizio 3.

Prova scritta di Calcolatori Elettronici del 19/01/2005

Esercizio 1 Lo standard IEEE/ANSI 754 per la rappresentazione dei numeri floating point (virgola mobile) su 4 **byte**, prevede che vengano utilizzati: un bit s di segno, 8 bit X_E che rappresentano l'esponente E con fattore di polarizzazione (anche noto come fattore di traslazione o *bias*), 23 bit X_F che rappresentano la mantissa dopo il punto decimale.

(a) Si scriva la rappresentazione del numero 1, secondo questo standard, in esadecimale. In un certo istante il registro EAX contiene questo valore e viene eseguita l'istruzione:

```
shrl    $1, %eax
```

(b) Quale è il contenuto di EAX, in esadecimale, dopo questa istruzione?

(c) Quale è il numero floating point rappresentato dal valore corrente (ovvero dopo l'esecuzione dell'istruzione `shrl`) di EAX?

Esercizio 2 Viene eseguito il seguente codice:

```
xorb      $-2, %al
shlb      $1, %al
jbe       <indirizzo>
```

Per quali valori iniziali di AL il salto condizionale della terza istruzione avviene?

Esercizio 3 Si realizzi una rete con ingresso X su 4 bit e uscita z su un bit che quando $X \in \{0, 1, 2, 3, 4, 5, 7, 8, 9, 11, 13\}$ allora imposta $z = 1$, altrimenti $z = 0$. Si scriva z in forma minima SP (somma di prodotti) e forma minima PS (prodotti di somme).

Sol. Prova scritta di Calcolatori Elettronici del 19/01/2005

Esercizio 1 Si ha $1 = +1.0 \cdot 2^0$. Quindi:

- $s = 0$;
- $X_E = 0 + 2^{(K-1)} - 1 = 2^7 - 1 = 01111111_2$;
- $X_F = 0$ perché dopo il punto decimale ci sono tutti zeri.

Il numero quindi è:

$$0011\,1111\,1000\,0000\,0000\,0000\,0000\,0000_2 = \mathsf{3F\,80\,00\,00}_{16}$$

Dopo l'operazione di shift right di un bit, il valore di EAX diverrà:

$$0001\,1111\,1100\,0000\,0000\,0000\,0000\,0000_2 = \mathsf{1F\,C0\,00\,00}_{16}$$

Il numero floating point da esso rappresentato ha: bit di segno $s = 0$, esponente $E = 2^6 - 1 - (2^7 - 1) = -2^6$ e mantissa $F = 1.1$ perché nell'operazione di shift right un bit è entrato in mantissa. Il numero quindi è:

$$1.1_2 \cdot 2^{-64} = 1.5 \cdot 2^{-64} \approx 8.131516 \cdot 10^{-20}$$

Esercizio 2 Sia $X = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$ il valore iniziale incognito del registro AL.

Il valore -2, operando byte sorgente immediato della prima istruzione, è rappresentato da 1111 1110. Quindi, ricordando che lo XOR con 1 equivale ad una negazione mentre lo XOR con 0 lascia inalterato il bit, il contenuto di AL dopo la prima istruzione sarà $(\overline{x_7}, \overline{x_6}, \overline{x_5}, \overline{x_4}, \overline{x_3}, \overline{x_2}, \overline{x_1}, x_0)$.

La seconda istruzione è uno shift left di un bit quindi il contenuto di AL sarà $(\overline{x_6}, \overline{x_5}, \overline{x_4}, \overline{x_3}, \overline{x_2}, \overline{x_1}, x_0, 0)$, ed il carry flag conterrà il valore di $\overline{x_7}$, in accordo al significato dell'istruzione di shift.

Il salto condizionale `jbe` salta se il carry flag è 1 oppure se lo zero flag è 1. Il salto avviene quindi se :

- $\overline{x_7} = 1$ e quindi $x_7 = 0$, oppure se
- $(\overline{x_6}, \overline{x_5}, \overline{x_4}, \overline{x_3}, \overline{x_2}, \overline{x_1}, x_0, 0) = 0$ e quindi se $x_6 = 1$, $x_5 = 1$, $x_4 = 1$, $x_3 = 1$, $x_2 = 1$, $x_1 = 1$, $x_0 = 0$,

che in sintesi vuol dire $X \in \{0, 1, 2, \ldots, 126, 127, -2\}$.

Esercizio 3 Svolgendo le Mappe di Karnaugh si ottiene direttamente che:

- $z = \overline{x_3}\,\overline{x_2} + \overline{x_3}\,\overline{x_1} + \overline{x_3}\,x_0 + \overline{x_1}\,x_0 + \overline{x_2}\,\overline{x_1} + \overline{x_2}\,x_0$
- $z = (\overline{x_2} + \overline{x_1} + x_0)(\overline{x_3} + \overline{x_2} + \overline{x_1})(\overline{x_3} + \overline{x_1} + x_0)(\overline{x_3} + \overline{x_2} + x_0)$.

Prova scritta di Calcolatori Elettronici del 31/03/2005

Esercizio 1 Si realizzi unicamente con porte NAND una rete combinatoria con ingresso X su 3 bit e uscita z su un bit che quando $X \in \{0, 2, 4, 5, 6\}$ allora imposta $z = 1$, altrimenti $z = 0$.

Supponiamo che ogni porta NAND utilizzata introduca un ritardo di Δ dell'uscita rispetto all'ingresso. All'istante t avviene una transizione del segnale x_1. Dopo quanto tempo l'uscita z può considerarsi stabile? Dopo quanto tempo dopo una transizione dei segnali x_0 e x_2 l'uscita può considerarsi stabile?

Esercizio 2 Poggibonsi è un importante distretto industriale per la fabbricazione di utensili per la lavorazione del legno. Questi utensili sono realizzati da macchine a controllo numerico. Le macchine a controllo numerico sono capaci di eseguire delle operazioni in base alle **istruzioni** che vengono loro impartite. Queste istruzioni sono opportunamente codificate da **sequenze di byte**, in maniera identica a quanto accade per i processori. In particolare la macchina hexCube è capace di eseguire le seguenti operazioni:

Codifica (hex)	**Operazione**
01	deposita il pezzo corrente (se presente) e preleva prossimo pezzo dal magazzino
02 XX	trasla il pezzo corrente di X millimetri, dove X è l'intero **con segno** rappresentato dal byte XX
04 XX	ruota il pezzo corrente di X gradi, dove X è l'intero **con segno** rappresentato dal byte XX
08	taglia il pezzo
10	fresa il pezzo
20 YY	ripeti la prossima operazione per Y volte, dove Y è l'intero positivo rapprentato dal byte YY
altro	Codifica non valida. Non eseguire operazioni

Elencare le operazioni che avvengono quando viene eseguita la seguente sequenza di byte:

01 02 04 06 08 10 20 10 08 06 04 02 01

Esercizio 3 Realizzare un dispositivo dotato di 3 bit di uscita che produce la sequenza dei numeri primi minori di 8 (0 è considerato numero primo), ovvero la sequenza è $0, 1, 2, 3, 5, 7, 0, 1, \ldots$.

Il dispositivo è inoltre dotato di un segnale di /init attivo basso che quando marcato su valore 0 riporta il dispositivo al primo valore della sequenza, che è 0.

Per la realizzazione del dispositivo di possono utilizzare soltanto porte logiche elementari (AND, OR, NOT), porte XOR e NXOR e registri.

Sol. Prova scritta di Calcolatori Elettronici del 31/03/2005

Esercizio 1

X	x_2	x_1	x_0	z
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Dalla Mappe di Karnaugh si ottiene:

$$z = \overline{x_0} + x_2\,\overline{x_1} = \overline{\overline{\overline{x_0} + x_2\,\overline{x_1}}} = \overline{x_0\,\overline{x_2\,\overline{x_1}}}$$

Il segnale x_1 attraversa 3 porte, x_2 ne attraversa 2 e x_0 ne attraversa 1. Quindi dopo una transizione di x_1 l'uscita sarà stabile dopo 3Δ, 2Δ dopo x_2 e Δ dopo x_0. (Nota che una porta NAND con gli ingressi cortocircuitati è necessaria per negare x_1)

Esercizio 2 Dalla semplice interpretazione della sequenza di byte segue che le operazioni eseguite sono:

1. istr. 01. Prelievo pezzo;

2. istr. 02 04. Traslazione di 4 mm;
3. istr. 06. Codifica non valida. Nessuna operazione;
4. istr. 08. Tagliare il pezzo;
5. istr. 10. Fresare il pezzo;
6. istr. 20 10. Ripetere la prossima istruzione 16 volte;
7. istr. 08. Taglia il pezzo 16 volte;
8. istr. 06. Codifica non valida. Nessuna operazione;
9. istr. 04 02. Ruota il pezzo di 2 gradi;
10. istr. 01. Deposita il pezzo e preleva il prossimo.

Esercizio 3 Si realizza il dispositivo mediante una rete sequenziale sincrona. Gli stati sono 6, ognuno corrispondente ad un diverso valore dell'uscita. Una soluzione funzionante, benché meno interessante è, realizzare un contatore da 0 a 5 e, tramite una rete di uscita produrre il valore opportuno. In questo caso per le uscite non vengono prelevate dai registri e quindi sono meno stabili.

Se invece si sceglie di codificare gli stati con i valori stessi richiesti in uscita si ottiene un migliore comportamento della rete. Siccome segnale /init è attivo basso, è più conveniente implementare la rete per la legge di transizione degli stati come Prodotto di Somme (PS). Sia $\left(y_2^{(k)}, y_1^{(k)}, y_0^{(k)}\right)$ la codifica dello stato all'istante di marcatura k, con le mappe di Karnaugh si ottiene la seguente soluzione:

- $y_2^{(k+1)} = /\mathsf{init}\left(y_2^{(k)} + y_1^{(k)}\right)\left(\overline{y_2^{(k)}} + \overline{y_1^{(k)}}\right) y_0^{(k)} = /\mathsf{init}\left(y_2^{(k)} \oplus y_1^{(k)}\right) y_0^{(k)}$;
- $y_1^{(k+1)} = /\mathsf{init}\left(y_1^{(k)} + y_0^{(k)}\right)\left(\overline{y_1^{(k)}} + \overline{y_0^{(k)}}\right) = /\mathsf{init}\left(y_1^{(k)} \oplus y_0^{(k)}\right)$;
- $y_0^{(k+1)} = /\mathsf{init}\left(y_2^{(k)} + y_1^{(k)} + \overline{y_0^{(k)}}\right)\left(\overline{y_2^{(k)}} + \overline{y_1^{(k)}}\right)$.

Prova scritta di Calcolatori Elettronici del 26/04/2005

Esercizio 1 Sono date le seguenti istruzioni con le rispettive codifiche

Istruzione	Cod. (Hex)	Fetch		Indirizz.		Esecuzione	
		RD	WR	RD	WR	RD	WR
`movb $-1, (%eax)`	`C6 00 FF`						
`addw 4(%ebx), %dx`	`66 03 53 04`						
`sbbl %ebx, %ecx`	`19 D9`						
`outb %al, %dx`	`EE`						
`inw  %dx, %ax`	`66 ED`						

Per ogni istruzione si dica quanti byte vengono letti (RD) o scritti (WR) da/sul bus, specificando in quale fase dell'istruzione tale lettura/scrittura avviene.

Esercizio 2 Realizzare la rete che, dato un numero intero con segno X rappresentato su 4 bit, produce in uscita l'intero con segno $Z = \lfloor X/3 \rfloor$, ovvero l'approssimazione per difetto della divisione di X per 3.

Esercizio 3 È dato il circuito di Figura 14. Si osservi che i registri campionano i loro ingressi sui fronti di salita dei rispettivi segnali di controllo e che le porte tri-state abilitano l'uscita sul valore di 0 logico.

Si pilotino i segnali di controllo p_0, c_0, p_1, c_1, p_x e c_x in modo da scambiare i contenuti dei due registri x_0 e x_1. Si tenga presente che i valori logici iniziali dei segnali di controllo sono tutti 1.

Sol. Prova scritta di Calcolatori Elettronici del 26/04/2005

Esercizio 1 Per prima cosa ri rammenta che:

- durante la fase di Fetch, viene prelevata dalla memoria la codifica dell'istruzione e vengono quindi letti tanti byte quanto è lunga l'istruzione;
- durante la fase di Indirizzamento vengono prelevati gli operandi sorgente;
- durante la fase di Esecuzione viene eseguita l'istruzione vera e propria e, se necessario, vengono eseguiti dei cicli di scrittura.

Alla luce di questo si desume che i byte letti/scritti sono i seguenti:

Istruzione	Cod. (Hex)	Fetch		Indirizz.		Esecuzione	
		RD	WR	RD	WR	RD	WR
`movb $-1, (%eax)`	`C6 00 FF`	3	0	0	0	0	1
`addw 4(%ebx), %dx`	`66 03 53 04`	4	0	2	0	0	0
`sbbl %ebx, %ecx`	`19 D9`	2	0	0	0	0	0
`outb %al, %dx`	`EE`	1	0	0	0	0	1
`inw  %dx, %ax`	`66 ED`	2	0	2	0	0	0

Esercizio 2 La tabella di verità di tale rete combinatoria è la seguente

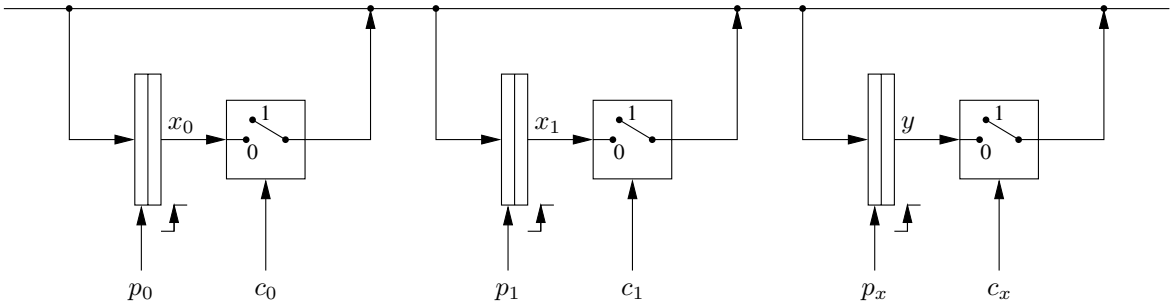

Figura 14: Schema - Esercizio 3

X	x_3	x_2	x_1	x_0	$Z = \lfloor X/3 \rfloor$	z_2	z_1	z_0
-8	1	0	0	0	-3	1	0	1
-7	1	0	0	1	-3	1	0	1
-6	1	0	1	0	-2	1	1	0
-5	1	0	1	1	-2	1	1	0
-4	1	1	0	0	-2	1	1	0
-3	1	1	0	1	-1	1	1	1
-2	1	1	1	0	-1	1	1	1
-1	1	1	1	1	-1	1	1	1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	1	1	1	0	0	1
4	0	1	0	0	1	0	0	1
5	0	1	0	1	1	0	0	1
6	0	1	1	0	2	0	1	0
7	0	1	1	1	2	0	1	0

Tramite Mappe di Karnaugh si trova che:

- $z_0 = \overline{x_3}\, x_2\, \overline{x_1} + x_3\, x_2\, x_1 + x_3\, \overline{x_2}\, \overline{x_1} + x_3\, x_2\, x_0 + \overline{x_3}\, \overline{x_2}\, x_1\, x_0$;
- $z_1 = x_3\, x_2 + x_3\, x_1 + x_2\, x_1$;
- $z_2 = x_3$.

Esercizio 3 Per scambiare i contenuti dei registri x_0 e x_1 si può usare il registro y come appoggio. Quindi si pilotano i segnali di controllo come riportato in Figure 15.

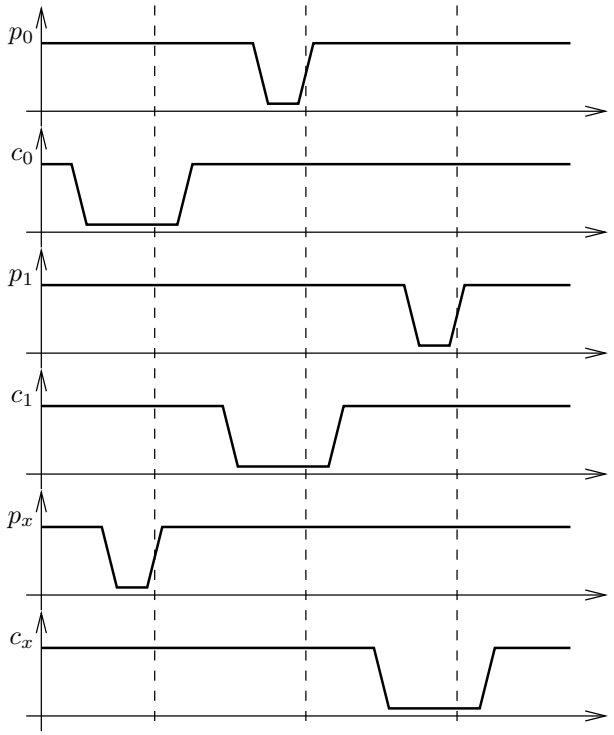

Figura 15: Temporizzazione segnali - Esercizio 3

Prova scritta straordinaria di Calcolatori Elettronici del 12/07/2005

Esercizio 1 In Tabella 11 si riportano alcune istruzioni presenti in memoria: nella terza colonna è riportata l'istruzione vera e propria, nella seconda la sua codifica e infine la prima colonna contiene l'indirizzo di memoria dove l'istruzione è memorizzata.

Ind. Memoria (Hex)	Codifica (Hex)	Istruzione
0x0000000E	A0 17 00 00 00	movb 0x00000017, %al
0x00000013	04 9C	addb $-100,%al
0x00000015	0F 83 0F 00 00 00	jnc 0x00000013
0x0000001B	34 F0	xorb $0xF0, %al

Tabella 11: Codice in memoria. Esercizio 1.

Appena prima della fase di fetch della prima istruzione del codice, il processore si trova nel seguente stato:

- il registro `EIP`, instruction pointer, contiene il valore `0x0000000E` (che è il valore corretto affinché possa avvenire il fetch all'indirizzo della prima istruzione).

Determinare il contenuto dei registri `EIP` ed `AL` al termine di ogni istruzione, fino a quando viene eseguita l'istruzione `xorb $0xF0, %al` inclusa.

Esercizio 2 Alcuni contatori possono essere impostati con valori iniziali decisi dall'utente. Questi dispositivi sono dotati anche di un ingresso numerico che permette di specificare il valore da cui iniziare a contare. Realizzarne uno su 2 bit che, quindi, è dotato dei seguenti segnali:

- un segnale di ingresso X su 2 bit, tramite il quale l'utente può specificare il valore iniziale;
- un segnale di ingresso `set` di un bit, attivo alto, che quando viene marcato su valore 1 imposta il prossimo valore della sequenza pari a X, altrimenti il contatore ignora X e quindi il prossimo valore sarà pari a quello precedente incrementato di 1;
- un segnale di uscita Z su 2 bit che produce la sequenza di uscita. Notare che, siccome il contatore è su 2 bit quando il valore dell'uscita è 3, il prossimo valore verrà riportato a 0;
- un segnale di marcatura p che determina l'avanzamento degli stati del contatore.

Per la realizzazione circuitale si possono utilizzare porte AND, OR, NOT, NAND, NOR, XOR e registri ad un bit.

Esercizio 3 Sul bus di comunicazione si verificano le segnalazioni riportate in Figura 16. Si risponda alle seguenti domande.

1. Quali operazioni avvengono negli intervalli $[t_0, t_2]$, $[t_2, t_4]$, $[t_4, t_6]$ e $[t_6, t_8]$?
2. Quale dispositivo scrive sul bus dati negli istanti t_1, t_3, t_5 e t_7?
3. Da cosa dipendono i tempi Δ_1 e Δ_2?

Sol. Prova scritta straordinaria di Calcolatori Elettronici del 12/07/2005

Esercizio 1 I valori dei registri sono quelli riportati in Tabella 12. Si riporta anche lo stato del Carry Flag (CF) che sarà utile ai fini della determinazione del flusso di esecuzione.

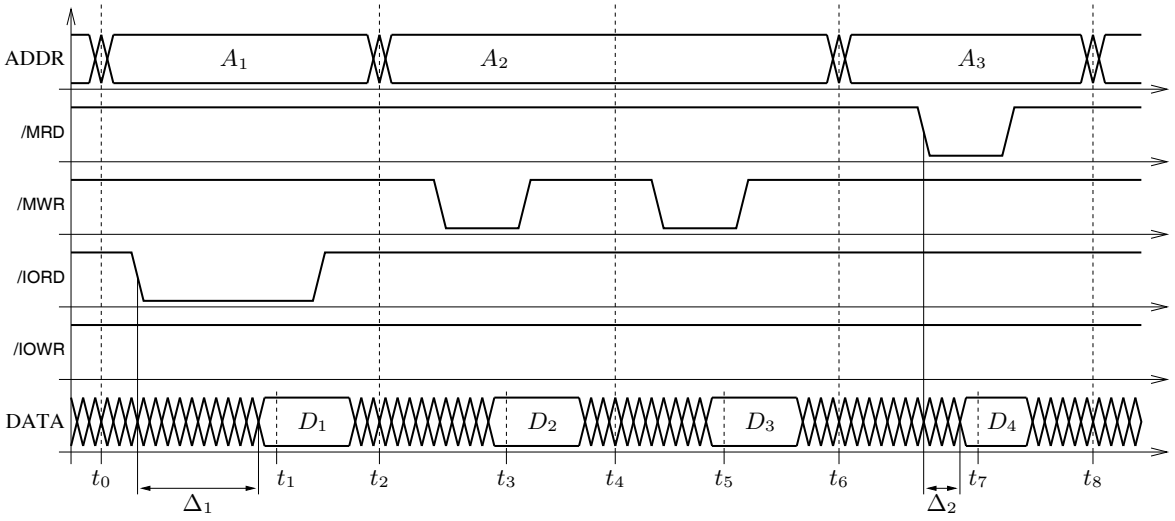

Figura 16: Esercizio 3 — Temporizzazioni.

Stato	`EIP` (Hex)	`AL` (Hex)	CF (bit)
Inizio	`00 00 00 0E`	unknown	unknown
dopo 1^a istr.	`00 00 00 13`	`0F`	unknown
dopo 2^a istr.	`00 00 00 15`	`AB`	0
dopo 3^a istr.	`00 00 00 13`	`AB`	0
dopo 2^a istr. (eseguita la seconda volta)	`00 00 00 15`	`47`	1
dopo 3^a istr. (eseguita la seconda volta)	`00 00 00 1B`	`47`	1
dopo 4^a istr.	`00 00 00 1D`	`B7`	1

Tabella 12: Tracing dei registri. Esercizio 1.

Esercizio 2 Quando il segnale set vale 0 il dispositivo si comporta come un normale contatore. Chiamate $y_0^{(k)}$ e $y_1^{(k)}$ le variabili di stato al k-esimo istante di marcatura, si avrà:

- $y_0^{(k+1)} = \overline{y_0^{(k)}}$;
- $y_1^{(k+1)} = y_0^{(k)} \oplus y_1^{(k)}$.

Se invece il segnale set vale 1 allora il prossimo stato sarà pari al valore dell'ingresso X. In questo caso si ha quindi:

- $y_0^{(k+1)} = x_0^{(k)}$;
- $y_1^{(k+1)} = x_1^{(k)}$.

Alla luce di questo, i due comportamenti possono essere combinati usando opportunamente il segnale set. La soluzione dell'esercizio infatti è:

- $y_0^{(k+1)} = \overline{\mathsf{set}}\,\overline{y_0^{(k)}} + \mathsf{set}\, x_0^{(k)}$;
- $y_1^{(k+1)} = \overline{\mathsf{set}}\,\left(y_0^{(k)} \oplus y_1^{(k)}\right) + \mathsf{set}\, x_1^{(k)}$.

Esercizio 3

1. Nell'intervallo $[t_0, t_2]$ avviene una lettura dallo spazio di I/O. Negli intervalli $[t_2, t_4]$ e $[t_4, t_6]$ avvengono due scritture consecutive in memoria allo stesso indirizzo. Infine nell'intervallo $[t_6, t_8]$ avviene una lettura da memoria.

2. Al tempo t_1 il bus dati viene scritto dal dispositivo di I/O in risposta al ciclo di lettura nei suoi confronti. Agli istanti t_3 e t_5 il bus dati viene scritto dal procesore. Infine al tempo t_7 il bus dati viene scritto dalla memoria in risposta al relativo ciclo di lettura.

3. Il tempo Δ_1 dipende dal tempo di risposta dell'interfaccia. Il tempo Δ_2 dipende dal tempo di risposta della memoria. Non sorprendentemente, si ha che $\Delta_1 \geq \Delta_2$ in quanto la memoria è tipicamente un dispositivo più veloce dei dispositivi di I/O.

Prova scritta di Calcolatori Elettronici del 06/09/2005

Esercizio 1 Ernesto, elettricista imbranato, deve collegare i 4 segnali di uscita di un contatore a 4 bit ad un display elettronico che deve visualizzare il numero relativo. Purtroppo, dopo aver effettuato il collegamento, la sequenza viasualizzata dal display è la seguente: ..., 12, 9, 13, 2, 6, 3, 7, 10, 14, 11, 15, 0, 4, 1, 5, 8, 12, 9, ... (senza sapere quale sia il valore iniziale!).

Che tipo di errore è stato commesso? Sareste in grado di risolverlo?

Esercizio 2 Si rammenta che la maschera di un dispositivo di I/O è una rete che produce 0 quando l'indirizzo fornito sul bus riguarda l'interfaccia, 1 quando l'indirizzo sul bus non riguarda l'interfaccia. Premesso questo, si realizzi una maschera che riconosce sia gli indirizzi nel intervallo `0x00F0`–`0x01FF` e che gli indirizzi nell'intervallo `0xFC00`–`0xFDFF`.

Inoltre, quanti sono gli indirizzi che devono essere riconosciuti?

Esercizio 3 Si scriva il codice Assembly che, a partire dall'indirizzo di memoria `0x00001000`, riempie la memoria con la successione di Fibonacci. Si ricorda che la successione di Fibonacci inizia con due numeri 1 e poi continua con la somma dei due numeri precedenti. I primi numeri della successione sono quindi: 1, 1, 2, 3, 5, 8, 13, 21 e così via.

Sol. Prova scritta di Calcolatori Elettronici del 06/09/2005

Esercizio 1 Sia X il numero che viene visualizzato e sia (x_3, x_2, x_1, x_0) la sua codifica (vedi Tabella 13).

Numero	x_3	x_2	x_1	x_0
12	1	1	0	0
9	1	0	0	1
13	1	1	0	1
2	0	0	1	0
6	0	1	1	0
3	0	0	1	1
7	0	1	1	1
10	1	0	1	0
14	1	1	1	0
11	1	0	1	1
15	1	1	1	1
0	0	0	0	0
4	0	1	0	0
1	0	0	0	1
5	0	1	0	1
8	1	0	0	0

Tabella 13: Sequenza visualizzata

Da una semplice ispezione delle codifiche si capisce che è stato scambiato l'ordine dei fili. È anche facile scoprire il modo in cui ripristinare i collegamenti. Infatti impostando:

- $y_0 = x_2$;
- $y_1 = x_0$;
- $y_2 = x_3$;
- $y_3 = x_1$,

Il valore Y eseguirà la sequenza del contatore a 4 bit.

Esercizio 2 Sia X l'indirizzo in ingresso alla maschera. Si ha che

- il segnale $/s_1 = x_{15} + x_{14} + x_{13} + x_{12} + x_{11} + x_{10} + x_9 + x_8 + \overline{x_7} + \overline{x_6} + \overline{x_5} + \overline{x_4}$ riconosce gli indirizzi nell'intervallo `0x00F0`–`0x00FF`;
- il segnale $/s_2 = x_{15} + x_{14} + x_{13} + x_{12} + x_{11} + x_{10} + x_9 + \overline{x_8}$ riconosce gli indirizzi nell'intervallo `0x0100`–`0x01FF`;
- il segnale $/s_3 = \overline{x_{15}} + \overline{x_{14}} + \overline{x_{13}} + \overline{x_{12}} + \overline{x_{11}} + \overline{x_{10}} + x_9$ riconosce gli indirizzi nell'intervallo `0xFC00`–`0xFDFF`.

Il segnale di select $/s$ che riconosce gli indirizzi di tutti gli intervalli è quindi dato dall'AND dei tre segnali (si ricordi che il select $/s$ è attivo basso) ovvero

$$/s = /s_1 \cdot /s_2 \cdot /s_3$$

Infine il numero di indirizzi è 16 (`0x00F0`–`0x00FF`) + 256 (`0x0100`–`0x01FF`) + 512 (`0xFC00`–`0xFFDF`) = 784.

Esercizio 3 Supponiamo per prima cosa di voler scrivere i numeri su long word (4 byte). Il codice per numeri su un byte o due è analogo.

Il problema può essere risolto dal seguente codice:

```
movl    $0x00001000, %ebx
movl    $1, (%ebx)
addl    $4, %ebx
movl    $1, %eax
movl    %eax, (%ebx)
addl    -4(%ebx), %eax
addl    $4, %ebx
movl    %eax, (%ebx)
jmp     <indirizzo di "addl -4(%ebx), %eax">
```

In questo codice il registro EAX mantiene sempre l'ultimo valore della sequenza. Il registro EBX, invece, funge da puntatore in memoria per la memorizzazione dei valori.

Prova scritta di Calcolatori Elettronici del 20/09/2005

Esercizio 1 Lo standard IEEE/ANSI 754 per la rappresentazione dei numeri *float* (su 4 byte), prevede che vengano utilizzati: un bit s di segno, 8 bit X_E che rappresentano l'esponente E con fattore di polarizzazione (anche noto come fattore di traslazione o *bias*), 23 bit X_F che rappresentano la mantissa dopo il punto decimale.

(a) Si scriva la rappresentazione del numero -21, secondo questo standard, in esadecimale. In un certo istante il registro EAX contiene questo valore esadecimale. In seguito viene eseguito il seguente codice:

```
movl    $0x00001000, %ebx
movl    %eax, (%ebx)
incl    %ebx
movw    (%ebx), %ax
shlw    $1, %ax
movw    %ax, (%ebx)
decl    %ebx
movl    (%ebx), %eax
```

(b) Si determini il contenuto del registro EAX al termine di ogni istruzione.

(c) Quale è il numero floating point rappresentato dal valore corrente (ovvero dopo l'esecuzione del codice) di EAX?

Esercizio 2 Una rete sequenziale sincrona riceve in ingresso, ad ogni istante di marcatura, un numero intero con segno X su 4 bit, rappresentato in complemento a 2. Questa rete è dotata di un unico segnale di uscita z che vale: 1 se gli ultimi due numeri in ingresso sono minori o uguali a zero; 0 altrimenti.

Realizzare la rete. Per la realizzazione della rete sincrona si possono utilizzare le porte logiche AND, OR, NOT, XOR, NXOR, NAND, NOR e registri ad un bit.

Esercizio 3 Realizzare una rete combinatoria che, dato un numero intero positivo X su 4 bit, ritorna 1 se il numero è compreso fra 0 e 9, 0 altrimenti. Si scriva la soluzione in forma minima PS e SP.

Se il numero fosse rappresentato su 5 bit, come cambierebbe la soluzione?