Architettura degli Elaboratori e delle Reti
Turno 2 – Prof. Federico PEDERSINI
Appello: 17 giugno 2004 – Prova scritta

Università degli Studi di Milano
Dip. Scienze dell'Informazione

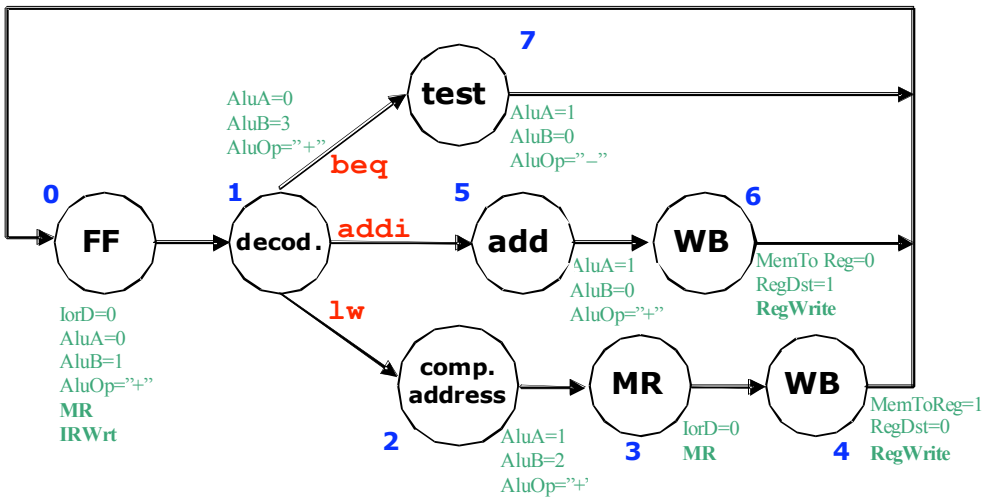
Soluzione

1. $A(B+C)+\overline{(\overline{B}+\overline{C})} = AB + AC + BC = AB(C+\overline{C}) + AC(B+\overline{B}) + BC(A+\overline{A}) =$

$= \overline{A}BC + ABC + AB\overline{C} + ABC + A\overline{B}C + ABC + \overline{A}BC = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC \quad c.v.d.$

2. Vedi schema moltiplicatore hardware – slides lezione 7

3.
STG:



STT:

Stato X	Ingressi: I			Uscite: Y = g(X)									
	lw	**addi**	**beq**	IorD	MR	MW	IR Wr	Reg Dst	Reg Write	Mem 2Reg	ALU srcA	ALU srcB	ALU Op
0	**1**	**1**	**1**	0	1	0	1	x	0	x	0	01	0 (+)
1	**2**	**5**	**7**	x	0	0	0	x	0	0	0	11	0 (+)
2	**3**	x	x	x	0	0	0	x	0	x	1	10	0 (+)
3	**4**	x	x	1	1	0	0	x	0	x	x	x	x
4	**0**	x	x	x	0	0	0	0	1	1	x	x	x
5	x	**6**	x	x	0	0	0	x	0	x	1	00	0 (+)
6	x	**0**	x	x	0	0	0	1	1	0	x	x	x
7	x	x	**0**	x	0	0	0	x	0	x	1	00	1 (–)

Codifica STT:

Stato X $x_2x_1x_0$	I = (i_1,i_0)			Uscite: Y = g(X)									
	00	**01**	**10**	IorD	MR	MW	IR Wr	Reg Dst	Reg Write	Mem 2Reg	ALU srcA	ALU srcB	ALU Op
000	**001**	**001**	**001**	0	1	0	1	x	0	x	0	01	0
001	**010**	**101**	**111**	x	0	0	0	x	0	0	0	11	0
010	**011**	x	x	x	0	0	0	x	0	x	1	10	0
011	**100**	x	x	1	1	0	0	x	0	x	x	x	x
100	**000**	x	x	x	0	0	0	0	1	1	x	x	x
101	x	**110**	x	x	0	0	0	x	0	x	1	00	0
110	x	**000**	x	x	0	0	0	1	1	0	x	x	x
111	x	x	**000**	x	0	0	0	x	0	x	1	00	1

N.B. I segnali di controllo dei MUX, se non significativi in uno stato, possono essere dichiarati indifferenti, mentre i comandi di scrittura registri vanno sempre specificati.

Copyright: N.A. Borghese, F. Pedersini – DSI, UniMI, 2004

Circuiti:

Funzioni stato prossimo: X* = f(X,I)	Funzioni uscita: Y = g(X)
$x_0^* = \overline{x_0}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}\,\overline{x_2}(i_0 + i_1) + \overline{x_0}\,\overline{x_2}\,i_0\,i_1$ $x_1^* = (\overline{x_0}\,\overline{x_1}\,\overline{x_2} + \overline{x_0}\,\overline{x_1}\,x_2)\,\overline{i_0}\,\overline{i_1} + x_0\,\overline{x_1}\,x_2\,i_0\,\overline{i_1}$ $x_2^* = x_0\,\overline{x_1}(i_0 + i_1) + x_0 x_1\,\overline{x_2}\,\overline{i_0}\,\overline{i_1}$	$IorD = x_0 x_1 \overline{x_2}$ $\quad RegWrite = \overline{x_0 x_2}$ $MR = x_2(x_0 x_1 + \overline{x_0}\,\overline{x_1})$ $\quad MemtoReg = \overline{x_1} x_2$ $MW = 0$ $\quad ALUsrcA = x_0 x_2 + x_1$ $IRWr = \overline{x_0 x_1 x_2}$ $\quad ALUsrcB = \{b_0 = \overline{x_1}\,\overline{x_2} \quad b_1 = \overline{x_2}(x_1 + x_0)\}$ $RegDst = x_1$ $\quad ALUop = x_1 x_2$

4. Essendo una cache 2-associativa di capacità C = 4 kByte, ogni blocco avrà capacità di 2 kB. Con linee contenenti 16 parole di 4 byte, oggni blocco è costituito da 2048 / (16*4) = 32 linee → campo index: 5 bit.
La memoria principale è di 1 Mbyte → gli indirizzi saranno composti di $\log_2(1M=2^{20})$= 20 bit.
L'indirizzo sarà così utilizzato per l'indirizzamento in cache:

tag (20 – 5 – 4 – 2 = **9** bit)	**index** (**5** bit)	offset di parola (**4** bit)	offset di byte (**2** bit)
← Indirizzo MP (20 bit) →			

Per lo schema della cache con circuiti di lettura/scrittura, vedi slides lezione 29.

6. Codice Assembly (tenendo conto che, in linguaggio macchina, l'istruzione **`mul`** è una pseudoistruzione e viene ottenuta con **`mult`** e quindi **`mflo`**:

```
      # Suppongo n scritto in $t6, uso $t5 per i
      add $t4, $zero, $zero          # azzera t4
      lui $t4, 2                     # ora t4 = 2^17
      add $t5, $zero, $zero          # azzera t5 (i)
      slt $t7, $t5, $t6
ciclo_for:
      beq $t7, $zero, end_for        # se not($t5<$t6), esci dal ciclo
      add $t0, $t1, $t4
      mul $t0, $t5
      mflo $t3
      j ciclo_for
end_for:
```

Linguaggio macchina (scelgo arbitrariamente di partire dall'indirizzo 300)

300:	0	0	0	11	0	32	→	000000 00000 00000 01011 00000 010000
304:	15	0	11	2			→	001111 00000 01011 0000000000000010
308:	0	0	0	12	0	32	→	000000 00000 00000 01100 00000 010000
312:	0	12	13	14	0	42	→	000000 01100 01101 01110 00000 0101010
316:	4	14	0	+4			→	000100 01110 00000 0000000000000011
320:	0	8	11	7	0	32	→	000000 01000 01011 00111 00000 100000
324:	0	7	12	0	0	24	→	000000 00111 01100 01010 00000 011000
328:	0	0	0	10	0	18	→	000000 00111 01100 01010 00000 011000
332:	2	316/4 = 79					→	000010 00000000000000000001001111
336:	…							

7. Codice: **0 → 010; 1 → 101**.
a) la distanza minima del codice, è la distanza di Hamming tra le uniche 2 parole del codice → $d_{MIN} = 3$
b) capacità di rivelazione: $r = d_{MIN} - 1 = 2$; capacità di correzione: $t = (d_{MIN} - 1) / 2 = 1$

8.

$$Speed - up = \frac{1}{1 - f_M + \frac{f_M}{s_M}} = \frac{1}{0{,}2 + \frac{0{,}8}{16}} = 4$$

$$Speed - up_{MAX} = \lim_{s_M \to \infty}(Speed - up) = \frac{1}{1 - f_M} = 5$$

9. $102_{10} = = 1100110_2$; $0{,}375_{10} = = 0{,}011_2$
$102{,}375 = 1100110{,}011 = 1{,}100110011 \cdot 2^6$

102,375	0 | 1000 0101 | 100 1100 1100 0000 0000 0000

Copyright: N.A. Borghese, F. Pedersini – DSI, UniMI, 2004