

Esercizi Architettura degli Elaboratori.

Es. 1 (Pipeline)

1.1 Esercizio 1

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

ISTR. 1 2 3 4 S 6 7 8 9 10 11

addl %eax, %ebx	F D E M S
movl \$4, %ecx	F D E M S
subl %ebx, %ecx	F D D D E M S
movl \$4, %edx	F F F F D E M S

Es 2 (Pipeline)

1.2 Esercizio 2

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

ISTR. 1 2 3 4 S 6 7 8 9 10 11

ciclo: addl %eax, %ebx	F D E M S	F D D E
movl \$4, %ecx	F D E M S	F F D
subl %eax, %edx	F D E M S	F
movl \$6, %ebx	F D D E M S	
jmp ciclo	F F D E M S	

Es 3 (Pipeline)

1.3 Esercizio 3

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1. Si ipotizzi che il salto avvenga. Si ignorino le tecniche del Delay Slot e della Branch Prediction. I commenti #yes e #no indicano se il salto avviene o meno.

ISTR. 1 2 3 4 5 6 7 8 9 10 11 12 13

inizio: inc %ebx	F D E M S	F D E M S
movl %ecx, %edx	F D E M S	F D E M
cmpl %eax, 0x86FF	F D E M S	F D E
jne inizio <i># yes</i>	F D D D D E M S F D	
movl %ecx, %edx	F F F F	F

Es 4 (Pipeline)

1.4 Esercizio 4

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e che il salto non avvenga.

ISTR. 1 2 3 4 5 6 7 8 9 10 11

START: subl %eax, %ebx	F D E M S
jz START <i># no</i>	F D D D D E M S
subl %ebx, %ecx	F F F F D E M S
movl %edx, %eax	F D E M S

Es 5 (Pipeline)

1.5 Esercizio 5

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e che il salto non avvenga.

ISTR.
ciclo: oddl %eax, %ebx
movl %edx, %ecx
subl %ebx, %ecx
jz ciclo # no
movl %ecx, %edx

1	2	3	4	S	6	7	8	9	10	11	12	13	14	15
F	D	E	M	S										
F	D	E	M	S										
F	D	D	D	E	M	S								
F	F	F	D	D	D	E	M	S						
F	F	F	D	E	M	S								

Es 1 (Codifica)

Somme due numeri binari in compl. a 2 e convertiti in $_{10}$

$$A = 100110_2$$

$$\begin{array}{r} 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \\ + 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \\ \hline 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \end{array} \rightarrow 011010_2 = -26_{10}$$

$$\begin{array}{r} 1 \quad 1 \quad 0 \quad 1 \\ + 1 \quad 1 \quad 1 \quad 1 \\ \hline 1 \quad 1 \quad 0 \quad 0 \end{array} \rightarrow 0011_2 = -3_{10}$$

$$-26 + (-3) = -29_{10}$$

$$\begin{array}{r} 1 \quad 1 \quad 1 \\ 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \\ + 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \\ \hline 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \end{array} = 011101 = -29_{10}$$

Es 2 (Codifica)

Converti 13 EF3C8B in virgola mobile a base 10

0 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1
 A B C D E F G H

0 0 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 0 1 0 1 1
 segno esponente mantissa

$$\text{mantissa} = 1 \cdot \frac{1}{2^0} + \frac{1}{2^1} \cdot \frac{1}{2^2} + \frac{1}{2^4} \cdot \frac{1}{2^5} + \frac{1}{2^6} \cdot \frac{1}{2^7} + \frac{1}{2^8} \cdot \frac{1}{2^9} + \frac{1}{2^{10}} \cdot \frac{1}{2^{11}} + \frac{1}{2^{12}} \cdot \frac{1}{2^{13}} + \frac{1}{2^{14}} \cdot \frac{1}{2^{15}} + \frac{1}{2^{16}} \cdot \frac{1}{2^{17}} + \frac{1}{2^{18}} \cdot \frac{1}{2^{19}} + \frac{1}{2^{20}} \cdot \frac{1}{2^{21}} + \frac{1}{2^{22}} \cdot \frac{1}{2^{23}}$$

$\hookrightarrow 1,863035$

$$\text{esponente } 00100111 \rightarrow 1 + 2 + 4 + 32 = 39$$

$$39 - 127 = -88$$

$$+ 2^{-88} \cdot 1,863035 = 6,0351 \cdot 10^{-27}$$

Es 1 (CACHE)

3) Si consideri una CPU dotata di memoria cache 2-associativa di 8K parole con 64 parole per blocco. Questa CPU è collegata ad una memoria RAM da 8M parole.

Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO ed ETICHETTA in cui questo indirizzo può essere suddiviso. Motivare la risposta con un opportuno schema.

CACHE 2 - Set

$$\text{dim. Cache} = 8K = 2^3 \cdot 2^{10} = 2^{13} = 8192 \text{ parole}$$

$$\text{parole} = 64 = 2^6$$

$$\text{RAM} = 8M \rightarrow 2^3 \cdot 2^{20} = 2^{23}$$

bit per individuare la RAM = $\log_2 2^{23} = 23$ bit

$$\text{Parole} = \log_2 64 = \log_2 2^6 = 6 \text{ bit}$$

$$\text{Set} = \log_2 \frac{2^{13}}{2^6} = 2^6 = 6 \text{ bit}$$

$$\text{Etichette} = \log_2 \frac{\frac{2}{2^6}}{2^6} = 2^{11} \rightarrow 11 \text{ bit}$$

etichette = 11 bit	Set = 6 bit	parole = 6 bit
--------------------	-------------	----------------

Si assume che la cache appena descritta sia utilizzata per i dati, che sia inizialmente vuota e che utilizzi un algoritmo di sostituzione dei blocchi di tipo LRU (sostituzione dell'elemento meno utilizzato di recente). La CPU esegue un programma che accede in sequenza a tutti gli elementi di un array di 8320 parole (ogni elemento ha le dimensioni di una parola) che è memorizzato a partire dall'indirizzo 0. Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 5 volte.

Si assume che il tempo di accesso alla cache sia di 1T e che il tempo di accesso alla memoria sia di 10T (entrambi i tempi si riferiscono alla lettura di una parola).

Calcolare il rapporto (fattore di miglioramento) tra il sistema in presenza di cache e in assenza di cache per l'esecuzione di questo programma.

alg = LRU

T_{RAM} = 10T

dim. array = 8320

T_{CACHE} = 1T

ciclo = 5

$$T_{senza\ cache} = 8320 \cdot 5 \cdot 10 = 416000 \text{ T}$$

$$\frac{8192}{64} = 128 \text{ blocchi}$$

$$\frac{8320}{64} = 130 \text{ blocchi}$$

$$1^{\circ} \text{ ciclo} = 130 \text{ miss}$$

$$2^{\circ} - 5^{\circ} \text{ ciclo} = (2 \cdot 2 + 2) \cdot 4 = 24 \text{ miss}$$

$$124 \text{ miss}$$

$$\text{Totale blocchi letti da RAM} = 154 \text{ blocchi}$$

$$\text{Totale parole lette da RAM} = 154 \cdot 64 = 9856 \text{ parole}$$

$$\text{Totale parole lette da CACHE} = 8320 \cdot 5 = 9856 = 31744 \text{ parole}$$

$$\text{Tempo totale di accesso con CACHE} = \text{Totale parole RAM} + \text{tot parole}$$

$$\text{CACHE} = 9856 \cdot 10 + 31744 \cdot 1 = 130304 \text{ T}$$

$$\text{Fattore di moltiplicazione} = \frac{416000\text{T}}{130304\text{T}} = 3,2\text{T}$$

Esercizio 1 (Microistruzioni)

Scrivi il Fetch

- 1) PC_{out}, MAR_{IN}, READ, SELECT[4], ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}

Esercizio 2 (Microistruzioni)

INC %EAX

- 1) PC_{out}, MAR_{IN}, READ, SELECT[4], ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}
- 4) EAX_{out}, SELECT[0], CB, ADD, Z_{IN}
- 5) Z_{out}, EAX_{IN}, END

Esercizio 3 (Microistruzioni)

INC variabile

- 1) PC_{out}, MAR_{IN}, READ, SELECT[4], ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}, END
- 4) IR_{imm.field}_{out}, MAR_{in}, READ
- 5) WMFC
- 6) MDR_{out}, SELECT₀, CB, ADD, Z_{IN}
- 7) Z_{out}, MDR_{IN}, WRITE
- 8) WMFC
- 9) END

Esercizio 2 (CACHE)

Esercizio 1 Prima parte

Si consideri una CPU dotata di memoria cache 4-associativa di 8K parole con 64 parole per blocco. Questa CPU è collegata ad una memoria RAM da 8M parole.

Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO ed ETICHETTA in cui questo indirizzo può essere suddiviso. Motivare la risposta con un opportuno schema.

CACHE 4 set

$$\text{dim. Cache} = 8\text{K} = 2^3 \cdot 2^{10} = 2^{13} = 8192 \quad \text{parole} = 64$$

$$\text{dim. RAM} = 8\text{M} = 2^3 \cdot 2^{20} = 2^{23}$$

$$\text{dimensione indirizzamento RAM} = \log_2 2^{23} = 23 \text{ bit}$$

$$\text{dim. Parola} = \log_2 2^6 = 6 \text{ bit}$$

$$\text{dim. set} = \log_2 \frac{2^{13}}{2^6 \cdot 2^2} = 5 \text{ bit}$$

$$\text{ETICHETTA} = \underbrace{23 - 6 - 5}_{\downarrow \text{bit indirizzamento RAM}} = 12 \text{ bit}$$

etichetta = 12 bit	set = 5 bit	parole = 6 bit
--------------------	-------------	----------------

Esercizio 1 Seconda parte

Si assume che la cache appena descritta sia utilizzata per i dati, che sia inizialmente vuota e che utilizzi un algoritmo di sostituzione dei blocchi di tipo LRU (sostituzione dell'elemento meno utilizzato di recente). La CPU esegue un programma che accede in sequenza a tutti gli elementi di un array di 8320 parole (ogni elemento ha le dimensioni di una parola) che è memorizzato a partire dall'indirizzo 0. Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 5 volte. Si assume che il tempo di accesso alla cache sia di 1T e che il tempo di accesso alla memoria sia di 10T (entrambi i tempi si riferiscono alla lettura di una parola). Calcolare il rapporto (fattore di miglioramento) tra il sistema in presenza di cache e in assenza di cache per l'esecuzione di questo programma.

$$\text{dim. Array} = 8320 \text{ parole} \quad \frac{8320}{64} = 130 \text{ blocchi}$$

$$\text{dim. Cache} = 8192 \text{ parole} \quad \frac{8192}{64} = 128 \text{ blocchi}$$

$$\text{alp. LRU} \quad \text{cicli} \rightarrow 5 \quad T_{CACHE} = 1T \quad T_{RAM} = 10T$$

$$T_{SENZA\ CACHE} = 8320 \cdot 10 \cdot 5 = 416000 \text{ T}$$

$$1^\circ \text{ Ciclo} = 130 \text{ Miss}$$

$$2^\circ - 5^\circ \text{ Ciclo} = (2 \cdot 4) + 2 = 10 \cdot 4 = 40 \text{ Miss}$$

$$\text{Totale blocchi letti dalla RAM} = 170 \text{ blocchi}$$

$$\text{Totale parole lette dalla RAM} = 170 \cdot 64 = 10880 \text{ parole}$$

$$\text{Totale parole lette dalla CACHE} = 8320 \cdot 5 - 10880 = 30720 \text{ parole}$$

$$\text{Tempo totale di accesso con CACHE} = 10880 \cdot 10T + 30720 = 139520T$$

$$\text{Fattore miglioramento} = \frac{416000}{139520} = 2,98$$

Es 3 (CACHE)

Esercizio 2

Un programma consiste in un totale di 300 istruzioni e contiene un ciclo di 50 istruzioni che è eseguito 15 volte. Il processore contiene una cache. Prelevare ed eseguire un'istruzione che si trova in memoria principale richiede 20 unità di tempo. Se l'istruzione si trova nella cache, prelevare ed eseguire l'istruzione richiede solo 2 unità di tempo. Ignorando gli accessi dovuti al prelievo degli operandi, calcolare il rapporto tra il tempo di esecuzione del programma senza la cache e il tempo di esecuzione con la cache. Questo rapporto è chiamato accelerazione (speedup) dovuta all'uso della cache. Assumere che la cache sia inizialmente vuota, che sia sufficientemente grande da contenere il ciclo e che il programma inizi con tutte le istruzioni presenti in memoria principale.

300 istruzioni → ciclo di 50 istruzioni
 ↓
 eseguito 15 volte

$$\begin{aligned} T_{RAM} &= 20T \\ T_{CACHE} &= 2T \end{aligned}$$

$$T_{SENZA\,CACHE} = 250 \cdot 20 + 50 \cdot 15 \cdot 20 = 20000$$

$$\hookrightarrow 250 \text{ (istruzioni fuori ciclo} \cdot T) \quad \hookrightarrow 50 \cdot 15 \\ \text{(istruzioni nel ciclo} \cdot \text{ciclo} \cdot T)$$

$$T_{CACHE} = 300 \cdot 20 + 50 \cdot 2 \cdot 14 = 7400$$

il primo caricamento in cache avviene su tutto il blocco istruzioni
 Una volta caricate dobbiamo calcolare il tempo delle altre 14 esecuzioni quindi $50 \cdot 14 \cdot T$

$$\text{Speed up} = \frac{20000}{7400} = 2,7$$

Es 4 (CACHE)

Esercizio 3

a) Supporre che il tempo di esecuzione per un programma sia proporzionale al tempo di prelievo delle istruzioni. Assumere che prelevare un'istruzione dalla cache richieda 1 unità di tempo, ma che prelevarla dalla memoria principale richieda 10 unità di tempo. Inoltre assumere che un'istruzione richiesta sia trovata nella cache con probabilità 0.96. Infine assumere che, se un'istruzione non è trovata nella cache, essa debba essere prima prelevata dalla memoria principale e inserita nella cache, poi prelevata dalla cache per essere eseguita. Calcolare il rapporto tra il tempo di esecuzione di un programma senza cache e il tempo di esecuzione con cache. Questo rapporto è chiamato accelerazione (speedup) dovuta alla presenza della cache.

b) Se la dimensione della cache è raddoppiata, assumere che la probabilità di non trovare una istruzione richiesta sia dimezzata. Ripetere la parte a per una cache con dimensione doppia.

$$T_{CACHE} = 1T$$

$$T_{RAM} = 10T$$

$$\% \text{ che istruzione sia in CACHE} = 0,96$$

$$\% \text{ che non sia in CACHE} = 0,04$$

$$T_m(0,96 \cdot 1) + (0,04 \cdot 11) = 1,4$$

essendo le possibili che non sia in CACHE
 abbiamo $10T_{mem} + 1T_{cache}$

$$a) \text{ Speedup} = \frac{10}{1,4} \approx 7,1$$

$$b) 0,04 / 2 = 0,02 \rightarrow \text{Raddoppiando la CACHE dimezziamo le prob}$$

di miss

$$T_m = (0,38 \cdot 1) + (0,02 \cdot 11) = 1,2$$

$$\text{speed up} = \frac{10}{1,2} = 8,3$$

Esempio 5 (CACHE)

Esercizio 4

Si consideri una memoria cache 4-set associativa della dimensione di 32 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 1Mbyte indirizzabile per byte. Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Cache 4-set

$$\text{dim. Cache} = 32K = 2^5 \cdot 2^{10} = 2^{15} = 32768 \text{ bit}$$

$$\text{dim. parola} = 1024 = 2^{10}$$

$$\text{dim. Mem} = 1M = 2^{20}$$

$$\text{dim. indirizzo RAM} = \log_2 2^{20} = 20 \text{ bit}$$

$$\text{dim. indirizzo CACHE} = \log_2 2^{15} = 15 \text{ bit}$$

$$\text{parola} = \log_2 2^{10} = 10 \text{ bit}$$

$$\text{blocco} = \log_2 \frac{2^{15}}{2^{10} \cdot 2^2} = 3 \text{ bit}$$

$$\text{etichetta}_n = 20 - 3 - 10 = 7 \text{ bit}$$

etichetta _n 7 bit	blocco 3 bit	parola 10 bit
---------------------------------	-----------------	------------------

$$\text{etichetta}_c = 15 - 3 - 10 = 2$$

← blocchi e parole naturalmente sono identici

e _c 26 bit	blocco 3 bit	parola 10 bit
--------------------------	-----------------	------------------

Esempio 6 (CACHE)

Esercizio 5

Si consideri una memoria cache 4-set associativa della dimensione di 16 Kbyte con 512 byte per blocco. La cache è collegata ad una memoria di 2Mbyte indirizzabile per byte.

Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Cache 4 set

$$\text{dim. Cache} = 16K = 2^4 \cdot 2^{10} = 2^{14} = 16384$$

$$\text{dim. RAM} = 2M = 2 \cdot 2^{20} = 2^{21}$$

$$\text{parola} = 5 \cdot 2 = 2^5$$

$$\text{bit indirizzo 12AM} = \log_2 2^{21} = 21$$

$$\text{parola} = \log_2 2 = 9 \text{ bit}$$

$$\text{blocco} = \log_2 \frac{2^{14}}{2^9 \cdot 2^2} = 3 \text{ bit}$$

$$\text{etichetta} = 21 - 9 - 3 = 9 \text{ bit}$$

etichetta 9 bit	blocco 3 bit	parola 9 bit
-----------------	--------------	--------------

$$\text{bit indirizzo CACHE} = \log_2 2^{14} = 14 \text{ bit}$$

$$\text{etichetta} = 14 - 9 - 3 = 2 \text{ bit}$$

e 2 bit	c 3 bit	blocco 3 bit	parola 9 bit
---------	---------	--------------	--------------

Es 4 (microistruzioni)

Esercizio 2

Elencare le micro istruzioni relative alla completa esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola e che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro:

MOVL (%EAX), %EBX

- 1) PC_{out}, MAR_{IN}, READ, SELECT₊, ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}
- 4) EAX_{out}, MAR_{IN}, READ
- 5) WMFC
- 6) MDR_{out}, EBX_{IN}, END

ES 5 (microistruzioni)

Esercizio 3

Elencare le micro istruzioni relative alla completa esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia **un solo BUS**, che l'istruzione sia composta da una sola parola e che (%EBX) rappresenti un metodo di indirizzamento indiretto a registro:

ADDL (%EBX), %EAX

- 1) PC_{out}, MAR_{IN}, READ, SELECT₊, ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}
- 4) EBX_{out}, MAR_{IN}, READ

5) WMFC, EAX OUT, V_{IN}

6) RDOUT, SELECT, ADD, Z_{IN}

7) ZOUT, EAX IN, END

ES 6 (Pipeline)

Esercizio 5

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e che il salto non avvenga.

ISTR

1 2 3 4 S 6 7 8 9 10 11 12 13 14 15

ciclo: addl %eax, %ebx F D E M S
movl %edx, %ecx F D E M S
subl %ebx, %ecx F D D D E M S
jz ciclo # no F F F D D D E M S
movl %ecx, %edx F F F D E M S

ES 7 (Pipeline)

Esercizio 7

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e si facciano le opportune ipotesi sul salto condizionale.

ISTR

1 2 3 4 S 6 7 8 9 10 11 12 13 14 15

init: movl %ecx, %edx F D E M S
addl \$4, %ebx F D E M S
cmpl 0x315FA, %ebx F D D D E M S
jnz init # no F F F D D D E M S
addl %eax, %ecx F F F D E M S

ISTR

1 2 3 4 S 6 7 8 9 10 11 12 13 14 15

init: movl %ecx, %edx F D E M S F D E M
addl \$4, %ebx F D E M S F D E
cmpl 0x315FA, %ebx F D D D E M S F D
jnz init # yes F F F D D D E M S F
addl %eax, %ecx F F F F

ES 1 (Mem. Virtuale)

Esercizio 3

Si assuma che un computer abbia memoria virtuale strutturata a pagine di 4Kbyte.

Si consideri un programma con un codice di 7.2 Kbyte che accede ciclicamente in sequenza a tutti gli elementi di un array di 1000 record in cui ogni campo è composto da 2 numeri interi ciascuno di 4 byte.

Quale deve essere la dimensione del *working set* perché si abbiano dei *page fault* solamente nella fase di caricamento del programma con esecuzione del primo ciclo di accesso agli elementi dell'array?

Quanti *page fault* si avrebbero durante l'esecuzione del programma ipotizzando che il *working set* abbia una pagina meno di quanto definito nel punto precedente e che il ciclo di accesso venga ripetuto 10 volte?

pag. mem. Virtuale = 4 K

programma = 7.2 K

array = 1000 record \rightarrow campi: 8

per contenere il programma servono 2 pagine e anche per i dati

a) Per avere page Fault solo nel caricamento serve un working set di 4 pagine

b)

c ₁	c ₂	d ₁
----------------	----------------	----------------

1° ciclo = 4 page Fault

2°-10° ciclo 2 page Fault per ciclo

page Fault totali = $4 + (9 \cdot 2) = 22$ page Fault

Es 3 (codifica)

$4 + \frac{3}{10}$ \rightarrow rappresento in virgola fissa 4 bit parte intera
5 bit parte frazionaria

$4_{10} \rightarrow 0100_2$

$$\begin{array}{r} \frac{3}{10} \\ \times 2 \\ \hline 0,6 \\ \times 2 \\ \hline 0,2 \\ \times 2 \\ \hline 0,4 \\ \times 2 \\ \hline 0,8 \\ \times 2 \\ \hline 0,6 \end{array} \rightarrow 01001_2$$

$4 + \frac{3}{10} = 0100101001_2 \rightarrow$ il risultato non è corretto perché dovendo troncare i bit non possiamo avere un risultato perfetto

Es 1 (Otl. reti combinatorie)

$O = F(A, B, C, D)$ $O = 0$ quando A è uguale a D o $B \neq C$

A B C D | O

Scrivere le tabelle delle verità

0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1

1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Es 2 (Ott. reti combinatorie)

$$f(a, b, c, d)$$

$$\text{ON-SET} = \{m_1, m_3, m_{11}, m_{13}\}$$

$$\text{OFF-SET} = \{m_0, m_2, m_5, m_8, m_9, m_{15}\}$$

a	b	c	d	o	m ₁	m ₃	m ₅	m ₈	m ₉	m ₁₁	m ₁₃	m ₀	m ₂	m ₄	m ₆	m ₇	m ₁₀	m ₁₂	m ₁₄	m ₁₅
m ₀	0	0	0	0	0	0	0	1	✓	m ₁ m ₃	0 0 - 1 I ₀	m ₄ m ₆ m ₁₂ m ₁₄	- 1 - 0							
m ₁	0	0	0	1	1	0	1	0	✓	m ₄ m ₆	0 1 - 0 ✓	m ₄ m ₁₂ m ₆ m ₁₄	- 1 - 0							
m ₃	0	0	1	0	0	0	0	1	✓	m ₄ m ₁₂	- 1 0 0 ✓									
m ₅	0	0	1	1	1	0	1	1	✓											
m ₈	0	1	0	0	-	0	0	1	0	m ₃ m ₇	0 - 1 1 I ₁									
m ₉	0	1	0	1	0	0	0	1	✓	m ₃ m ₁₁	- 0 1 1 I ₂									
m ₁₀	0	1	1	0	-	0	1	1	0	m ₆ m ₇	0 1 1 - I ₃									
m ₁₁	0	1	1	1	-	0	1	1	0	m ₆ m ₁₄	- 1 1 0 ✓									
m ₁₂	1	0	0	0	0	0	0	1	✓	m ₁₀ m ₁₁	1 0 1 - I ₄									
m ₁₃	1	0	0	1	0	0	0	1	✓	m ₁₀ m ₁₄	1 - 1 0 I ₅									
m ₁₄	1	0	1	0	-	0	1	1	0	m ₁₂ m ₁₃	1 1 0 - I ₆									
m ₁₅	1	0	1	1	1	0	0	1	✓	m ₁₂ m ₁₄	1 1 - 0 ✓									

$$I_0 \quad 1 \quad 1$$

$$I_0 \quad 1 \quad 1$$

$$I_1 \quad 1 \quad 1$$

$$I_2 \quad 1 \quad 1$$

$$I_2 \quad 1 \quad 1$$

$$I_6 \quad 1$$

X

↓

X

$$O = 0 0 - 1 + - 0 1 1 + 1 1 0 -$$

X

$$O = \bar{A} \bar{B} D + \bar{B} C D + A B \bar{C}$$

$$I_6 \quad 1$$

Es 3 (Ott. Combinatoria)

$$f(a, b, c, d)$$

$$\text{ON-SET} = \{m_3, m_4, m_8, m_9, m_{14}\}$$

$$\text{OFF-SET} = \{m_2, m_5, m_7, m_{12}, m_{10}, m_{15}\}$$

a	b	c	d	m	D	m ₀	m ₁	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₂	m ₁₄	m ₁₅
0	0	0	0	m ₀	-	m ₁	0	0	0	1	m ₀ m ₁	0 0 0 - ✓	m ₀ m ₁ m ₈ m ₉	- 0 0 - I ₆					
0	0	0	1	m ₁	-	m ₄	0	1	0	0	m ₀ m ₄	0 - 0 0 I ₀	m ₀ m ₈ m ₉ m ₁₄	- 0 0 -					
0	0	1	0	m ₂	0	m ₈	1	0	0	0	m ₀ m ₈	- 0 0 0 ✓							
0	0	1	1	m ₃	1						m ₁ m ₃	0 0 - 1 I ₁							
0	1	0	0	m ₄	1						m ₁ m ₃	- 0 0 1 ✓							
0	1	0	1	m ₅	0	m ₆	0	1	1	0	m ₄ m ₆	0 1 - 0 I ₂							
				m ₈	m ₉	m ₁₀	m ₁₂	m ₁₄	m ₁₅		m ₈ m ₉	1 0 0 - ✓							

0	1	1	0	m6	-	-	m5	1	0	0	1
0	1	1	1	m7	0						
1	0	0	0	m8	1	.	m11	1	0	1	1
1	0	0	1	m3	1	.	m13	1	1	0	1
1	0	1	0	m10	0		m14	1	1	1	0
1	0	1	1	m11	-	.					
1	1	0	0	m12	0						
1	1	0	1	m13	-	.	I ₀	I ₁	I ₂	I ₃	I ₄
1	1	1	0	m14	1	.	I ₅	I ₆	I ₇	I ₈	I ₉
1	1	1	1	m16	0		m3	1			

m4 1 1

m8

m5

m14

$$D = 0 - 00 + 00 - 1 + 01 - 0 + -011 + -1,0 + , -01$$

$$\overline{A}\overline{C}\overline{D} + \overline{A}\overline{B}D + \overline{A}B\overline{D} + \overline{B}C\overline{D} + BC\overline{D} + A\overline{C}D$$

Es 4 (OH. Combinatoria)

$$f(a, b, c, d) \quad \text{ON-SET} = \{m_1, m_3, m_5, m_{13}\} \quad \text{OFF-SET} = \{m_2, m_6, m_8, m_9, m_{15}, m_{16}\}$$

a	b	c	d	m	0	m0	0 0 0 0	m1	0 0 0 1	m2	0	m3	0 0 1 1	m4	0 1 0 0	m5	0	m6	0 1 1 0	m7	0 1 1 1	m8	0	m9	0	m10	-	m11	1 0 1 1	m12	1 1 0 0	m13	1 1 0 1	m14	1 1 1 0	m15	1 1 1 1	m16	0	m17	-	m18	0	m19	1 0 1 -	m20	m11	1 0 1 1	m21	m12	m13	1 1 0 -	m22	m12	m13	1 1 1 0	m23	m12	m14	1 1 1 0	m24	m12	m14	1 1 1 0	m25	m12	m14	1 1 1 0	m26	m12	m14	1 1 1 0	m27	m12	m14	1 1 1 0	m28	m12	m14	1 1 1 0	m29	m12	m14	1 1 1 0	m30	m12	m14	1 1 1 0	m31	m12	m14	1 1 1 0	m32	m12	m14	1 1 1 0	m33	m12	m14	1 1 1 0	m34	m12	m14	1 1 1 0	m35	m12	m14	1 1 1 0	m36	m12	m14	1 1 1 0	m37	m12	m14	1 1 1 0	m38	m12	m14	1 1 1 0	m39	m12	m14	1 1 1 0	m40	m12	m14	1 1 1 0	m41	m12	m14	1 1 1 0	m42	m12	m14	1 1 1 0	m43	m12	m14	1 1 1 0	m44	m12	m14	1 1 1 0	m45	m12	m14	1 1 1 0	m46	m12	m14	1 1 1 0	m47	m12	m14	1 1 1 0	m48	m12	m14	1 1 1 0	m49	m12	m14	1 1 1 0	m50	m12	m14	1 1 1 0	m51	m12	m14	1 1 1 0	m52	m12	m14	1 1 1 0	m53	m12	m14	1 1 1 0	m54	m12	m14	1 1 1 0	m55	m12	m14	1 1 1 0	m56	m12	m14	1 1 1 0	m57	m12	m14	1 1 1 0	m58	m12	m14	1 1 1 0	m59	m12	m14	1 1 1 0	m60	m12	m14	1 1 1 0	m61	m12	m14	1 1 1 0	m62	m12	m14	1 1 1 0	m63	m12	m14	1 1 1 0	m64	m12	m14	1 1 1 0	m65	m12	m14	1 1 1 0	m66	m12	m14	1 1 1 0	m67	m12	m14	1 1 1 0	m68	m12	m14	1 1 1 0	m69	m12	m14	1 1 1 0	m70	m12	m14	1 1 1 0	m71	m12	m14	1 1 1 0	m72	m12	m14	1 1 1 0	m73	m12	m14	1 1 1 0	m74	m12	m14	1 1 1 0	m75	m12	m14	1 1 1 0	m76	m12	m14	1 1 1 0	m77	m12	m14	1 1 1 0	m78	m12	m14	1 1 1 0	m79	m12	m14	1 1 1 0	m80	m12	m14	1 1 1 0	m81	m12	m14	1 1 1 0	m82	m12	m14	1 1 1 0	m83	m12	m14	1 1 1 0	m84	m12	m14	1 1 1 0	m85	m12	m14	1 1 1 0	m86	m12	m14	1 1 1 0	m87	m12	m14	1 1 1 0	m88	m12	m14	1 1 1 0	m89	m12	m14	1 1 1 0	m90	m12	m14	1 1 1 0	m91	m12	m14	1 1 1 0	m92	m12	m14	1 1 1 0	m93	m12	m14	1 1 1 0	m94	m12	m14	1 1 1 0	m95	m12	m14	1 1 1 0	m96	m12	m14	1 1 1 0	m97	m12	m14	1 1 1 0	m98	m12	m14	1 1 1 0	m99	m12	m14	1 1 1 0	m100	m12	m14	1 1 1 0	m101	m12	m14	1 1 1 0	m102	m12	m14	1 1 1 0	m103	m12	m14	1 1 1 0	m104	m12	m14	1 1 1 0	m105	m12	m14	1 1 1 0	m106	m12	m14	1 1 1 0	m107	m12	m14	1 1 1 0	m108	m12	m14	1 1 1 0	m109	m12	m14	1 1 1 0	m110	m12	m14	1 1 1 0	m111	m12	m14	1 1 1 0	m112	m12	m14	1 1 1 0	m113	m12	m14	1 1 1 0	m114	m12	m14	1 1 1 0	m115	m12	m14	1 1 1 0	m116	m12	m14	1 1 1 0	m117	m12	m14	1 1 1 0	m118	m12	m14	1 1 1 0	m119	m12	m14	1 1 1 0	m120	m12	m14	1 1 1 0	m121	m12	m14	1 1 1 0	m122	m12	m14	1 1 1 0	m123	m12	m14	1 1 1 0	m124	m12	m14	1 1 1 0	m125	m12	m14	1 1 1 0	m126	m12	m14	1 1 1 0	m127	m12	m14	1 1 1 0	m128	m12	m14	1 1 1 0	m129	m12	m14	1 1 1 0	m130	m12	m14	1 1 1 0	m131	m12	m14	1 1 1 0	m132	m12	m14	1 1 1 0	m133	m12	m14	1 1 1 0	m134	m12	m14	1 1 1 0	m135	m12	m14	1 1 1 0	m136	m12	m14	1 1 1 0	m137	m12	m14	1 1 1 0	m138	m12	m14	1 1 1 0	m139	m12	m14	1 1 1 0	m140	m12	m14	1 1 1 0	m141	m12	m14	1 1 1 0	m142	m12	m14	1 1 1 0	m143	m12	m14	1 1 1 0	m144	m12	m14	1 1 1 0	m145	m12	m14	1 1 1 0	m146	m12	m14	1 1 1 0	m147	m12	m14	1 1 1 0	m148	m12	m14	1 1 1 0	m149	m12	m14	1 1 1 0	m150	m12	m14	1 1 1 0	m151	m12	m14	1 1 1 0	m152	m12	m14	1 1 1 0	m153	m12	m14	1 1 1 0	m154	m12	m14	1 1 1 0	m155	m12	m14	1 1 1 0	m156	m12	m14	1 1 1 0	m157	m12	m14	1 1 1 0	m158	m12	m14	1 1 1 0	m159	m12	m14	1 1 1 0	m160	m12	m14	1 1 1 0	m161	m12	m14	1 1 1 0	m162	m12	m14	1 1 1 0	m163	m12	m14	1 1 1 0	m164	m12	m14	1 1 1 0	m165	m12	m14	1 1 1 0	m166	m12	m14	1 1 1 0	m167	m12	m14	1 1 1 0	m168	m12	m14	1 1 1 0	m169	m12	m14	1 1 1 0	m170	m12	m14	1 1 1 0	m171	m12	m14	1 1 1 0	m172	m12	m14	1 1 1 0	m173	m12	m14	1 1 1 0	m174	m12	m14	1 1 1 0	m175	m12	m14	1 1 1 0	m176	m12	m14	1 1 1 0	m177	m12	m14	1 1 1 0	m178	m12	m14	1 1 1 0	m179	m12	m14	1 1 1 0	m180	m12	m14	1 1 1 0	m181	m12	m14	1 1 1 0	m182	m12	m14	1 1 1 0	m183	m12	m14	1 1 1 0	m184	m12	m14	1 1 1 0	m185	m12	m14	1 1 1 0	m186	m12	m14	1 1 1 0	m187	m12	m14	1 1 1 0	m188	m12	m14	1 1 1 0	m189	m12	m14	1 1 1 0	m190	m12	m14	1 1 1 0	m191	m12	m14	1 1 1 0	m192	m12	m14	1 1 1 0	m193	m12	m14	1 1 1 0	m194	m12	m14	1 1 1 0	m195	m12	m14	1 1 1 0	m196	m12	m14	1 1 1 0	m197	m12	m14	1 1 1 0	m198	m12	m14	1 1 1 0	m199	m12	m14	1 1 1 0	m200	m12	m14	1 1 1 0	m201	m12	m14	1 1 1 0	m202	m12	m14	1 1 1 0	m203	m12	m14	1 1 1 0	m204	m12	m14	1 1 1 0	m205	m12	m14	1 1 1 0	m206	m12	m14	1 1 1 0	m207	m12	m14	1 1 1 0	m208	m12	m14	1 1 1 0	m209	m12	m14	1 1 1 0	m210	m12	m14	1 1 1 0	m211	m12	m14	1 1 1 0	m212	m12	m14	1 1 1 0	m213	m12	m14	1 1 1 0	m214	m12	m14	1 1 1 0	m215	m12	m14	1 1 1 0	m216	m12	m14	1 1 1 0	m217	m12	m14	1 1 1 0	m218	m12	m14	1 1 1 0	m219	m12	m14	1 1 1 0	m220	m12	m14	1 1 1 0	m221	m12	m14	1 1 1 0	m222	m12	m14	1 1 1 0	m223	m12	m14	1 1 1 0	m224	m12	m14	1 1 1 0	m225	m12	m14	1 1 1 0	m226	m12	m14	1 1 1 0	m227	m12	m14	1 1 1 0	m228	m12	m14	1 1 1 0	m229	m12	m14	1 1 1 0	m230	m12	m14	1 1 1 0	m231	m12	m14	1 1 1 0	m232	m12	m14	1 1 1 0	m233	m12	m14	1 1 1 0	m234	m12	m14	1 1 1 0	m235	m12	m14	1 1 1 0	m236	m12	m14	1 1 1 0	m237	m12	m14	1 1 1 0	m238	m12	m14	1 1 1 0	m239	m12	m14	1 1 1 0	m240	m12	m14	1 1 1 0	m241	m12	m14	1 1 1 0	m242	m12	m14	1 1 1 0	m243	m12	m14	1 1 1 0	m244	m12	m14	1 1 1 0	m245	m12	m14	1 1 1 0	m246	m12	m14	1 1 1 0	m247	m12	m14	1 1 1 0	m248	m12	m14	1 1 1 0	m249	m12	m14	1 1 1 0	m250	m12	m14	1 1 1 0	m251	m12	m14	1 1 1 0	m252	m12	m14	1 1 1 0	m253	m12	m14	1 1 1 0	m254	m12	m14	1 1 1 0	m255	m12	m14	1 1 1 0	m256	m12	m14	1 1 1 0	m257	m12	m14	1 1 1 0	m258	m12	m14	1 1 1 0	m259	m12	m14	1 1 1 0	m260	m12	m14	1 1 1 0	m261	m12	m14	1 1 1 0	m262	m12	m14	1 1 1 0	m263	m12	m14	1 1 1 0	m264	m12	m14	1 1 1 0	m265	m12	m14	1 1 1 0	m266	m12	m14	1 1 1 0	m267	m12	m14	1 1 1 0	m268	m12	m14	1 1 1 0	m269	m12	m14	1 1 1 0	m270	m12	m14	1 1 1 0	m271	m12	m14	1 1 1 0	m272	m12	m14	1 1 1 0	m273	m12	m14	1 1 1 0	m274	m12	m14	1 1 1 0	m275	m12	m14	1 1 1 0	m276	m12	m14	1 1 1 0	m277	m12	m14	1 1 1 0	m278	m12	m14	1 1 1 0	m279	m12	m14	1 1 1 0	m280	m12	m14	1 1 1 0	m281	m12	m14	1 1 1 0	m282	m12	m14	1 1 1 0	m283	m12	m14	1 1 1 0	m284	m12	m14	1 1 1 0	m285	m12	m14	1 1 1 0	m286	m12	m14	1 1 1 0	m287	m12	m14	1 1 1 0	m288	m12	m14	1 1 1 0	m289	m12	m14	1 1 1 0	m290	m12	m14	1 1 1 0	m291	m12	m14	1 1 1 0	m292	m12	m14	1 1 1 0	m293	m12	m14	1 1 1 0	m294

$I_2 \quad I_4 \quad I_7$

$m_1 \mid$

$$m_3 \mid \quad \rightarrow 0 = 00 - 1 + -011 + 110 -$$

$$m_{11} \mid \quad 0 = \overline{A} \overline{B} D + \overline{B} C D + A \overline{B} \overline{C}$$

$m_{13} \mid$

Esempio 1 (Codifica)

- ∞ in virgola mobile (IEEE754)

| 1 1 1 1 1 1 1 0 0 ...

La mantissa è composta da tutti 0 altrimenti da errore NaN. Esponente 12^{128} perché per definizione $2^{127} = \pm \infty$.

Esempio 2 (Codifica)

$4 + \frac{3}{10}$ in virgola fissa 4 bit per parte intera e 5 parte frazionaria

$$0100_2 \rightarrow 4_{10}$$

$$0100,01001_2$$

$$\begin{aligned} \frac{3}{10}_2 &\rightarrow 0,3_{10} \rightarrow 0,3 \cdot 2^0 & 0 \\ &\quad \left(\begin{array}{r} 0,6 \cdot 2^1 \\ 0,2 \cdot 2^0 \\ 0,4 \cdot 2^0 \\ 0,8 \cdot 2^1 \\ 0,6 \cdot 2^1 \\ 0,2 \cdot 2^0 \end{array} \right) & 1 \\ &\quad \vdots & 0 \end{aligned}$$

il numero non è preciso perché abbiamo un numero limitato di bit per la parte frazionaria essendo periodico

Esempio 3 (Codifico)

$$10001101_2$$

$$\text{MODULO} = 10001101_2 = 1 \cdot 2^0 + 1 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^7 = 128 + 8 + 4 + 1 = 141_{10}$$

$$\text{MODULO} \in \text{SEGUO} = 10001101_2 = -13_{10}$$

$$\begin{aligned} \text{COMPLEMENTO A DUE} &= 10001101 + \\ &\quad \underline{\quad 11111111} \\ &\quad \underline{10001100} \rightarrow 01110011_2 = -115_{10} \\ &\quad 64 + 32 + 16 + 2 + 1 \end{aligned}$$

$$\text{SOMMA} \quad 1011_2$$

$$\text{MODULO} = 1011_2 = 11_{10} \quad 11 + 141 = 152_{10}$$

$$\begin{array}{r} 10001101 + \\ 00001011 \\ \hline 10011000_2 = 152_{10} \end{array}$$

128 16 8

MODULO E SEGNO = 1011₂ = -3₁₀ -13 + (-3) = -16₁₀

$$\begin{array}{r} 1 \ 1 \ 1 \\ | \ 1 \ 0 \ | \\ \hline 0 \ 0 \ 1 \end{array}$$

$$1 \ 0 \ 0 \ 0 \ 0_2 \rightarrow 110000_2 = -16_{10}$$

COMPLEMENTO A DUE = $\begin{array}{r} 1 \ 1 \ 1 \\ | \ 0 \ 1 \ | \\ \hline 1 \ 1 \ 1 \end{array}_2 +$

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 \rightarrow 0101 = -5_{10} \\ 1 \ 1 \ 1 \ 1 \ 1 \\ | \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0_2 + \\ \hline 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \\ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \rightarrow 01111000_2 \rightarrow -120_{10} \\ 64 \ 32 \ 16 \ 8 \end{array}$$

$$-115 + (-5) = -120$$

Es 4 (Codifica)

33₁₀

- MODULO = 33 / 2 1 33₁₀ → 00100001₂

16 / 2	0
8 / 2	0
4 / 2	0
2 / 2	0
1 / 2	1
0	

uguale



- MODULO E SEGNO / COMPLEMENTO A DUE

SOMMA 110110 codificato in complemento a 2

$$\begin{array}{r} 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 2 + \\ | \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \\ \hline 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 2 \rightarrow 00101010_2 = -10_{10} \end{array}$$

$$33 - 10 = 23_{10}$$

$$\begin{array}{r} 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 + \\ | \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \\ \hline 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 2 \rightarrow +23_{10} \\ 16 \ 8 \ 4 \ 2 \ 1 \end{array}$$

Es 5 (Codifica)

Rappresenta su 6 bit (C. a 2) +24 -13

24 / 2	0	0 1 1 0 0 0 ₂	→ +24 ₁₀
12 / 2	0		
6 / 2	0		
3 / 2	1		
1 / 2	1		
0			

13 / 2	1
6 / 2	0
3 / 2	1
1 / 2	1
0	



$$01101_2 \rightarrow 10010 +$$

1
—

$$10011_2$$

① 10011
↳ estensione del segno

$$\begin{array}{r} 011000 \\ 110011 \\ \hline 001011 \end{array} \rightarrow +11_{10}$$

$$24 - 13 = 11_{10}$$

Interpretando 011000_2 e 110011_2 in modulo

la somma sarebbe $011000 +$

$$\begin{array}{r} 110011 \\ \hline 1001011 \end{array} \rightarrow 75_{10}$$

perche:

$$011000_2 = 22_{10}$$

$$110011_2 = 51_{10}$$

$$\text{Modulo e segno } 011000_2 \rightarrow 22_{10}$$

$$22 - 13 =$$

$$110011_2 \rightarrow -13_{10}$$

$$\begin{array}{r} 11000 + \\ 10011 \\ \hline 01011 \end{array} \rightarrow 11_{10}$$

→ quando c'è il cambio segno
il modulo e segno ha problemi
di logica.

Es 6 (Codifica)

$$04A0A001_{16} \rightarrow \text{virgola mobile singola precisione}$$

$$\underbrace{0000}_0 \quad \underbrace{0100}_4 \quad \underbrace{1010}_A \quad \underbrace{1010}_A \quad \underbrace{1010}_A \quad \underbrace{0000}_0 \quad \underbrace{0000}_0 \quad \underbrace{0001}_1$$

$$\underbrace{000001}_{S} \underbrace{001}_{E} \underbrace{0101010101010000000000000001}_{M}$$

$$S = (-1)^0 = 1$$

$$E = 00001001_2 = 9 - 127 = -118$$

$$M = 1,010101010101000000000000001_{2^{-13}}$$

$$\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \frac{1}{1024} +$$

$$N^o = (-1)^0 \cdot (1,01) \cdot 2^{-118} = 1 \cdot \left(1 + \frac{341}{1024} + 2^{-13}\right)^{-118} = \left(\frac{1365}{1024} + 2^{-13}\right)^{-118}$$

Es 6 (Codifica)

8 bit F2 e sommare in complemento e due 110101

$$11110010_2$$

$$\text{MODULO } 11110010_2 = 242_{10}$$

MODULO E SEGUO = 11110010₂ = -114₁₀

COMPLEMENTO A DOU = 11110010⁺

$$\begin{array}{r} 11110010 \\ 11111111 \\ \hline 11110001 \end{array} \rightarrow 00001110_2 \rightarrow -14_{10}$$

SUMMA (C. A DOU) = 11110010

$$\begin{array}{r} 11110010 \\ 11110101 \\ \hline 11111111 \\ 111100110 \end{array} \rightarrow 00011001_2 = -25_{10}$$

$$\begin{array}{r} 1111 \\ 110101 + \\ \hline 111111 \end{array}$$

$$111100 \rightarrow 001011_2 \rightarrow -11_2$$

$$-14 + (-11) = -25_{10}$$



Es 1 (04. Reti combinatorie)

$F(a, b, c, d)$

ON-SET = {m₃, m₄, m₈, m₉, m₁₄}

DC-SET = {m₂, m₅, m₇, m₁₂, m₁₀, m₁₅}

	a b c d	0	
m ₀	0 0 0 0	0	m ₂ 0010 ~
m ₁	0 0 0 1	0	m ₄ 0100
m ₂	0 0 1 0	-	m ₈ 1000
m ₃	0 0 1 1	1	m ₃ 0011
m ₄	0 1 0 0	1	m ₅ 0101
m ₅	0 1 0 1	-	m ₉ 1001
m ₆	0 1 1 0	0	m ₁₂ 1100
m ₇	0 1 1 1	-	m ₁₄ 1110
m ₈	1 0 0 0	1	m ₇ 0111
m ₉	1 0 0 1	1	m ₁₄ 1110
m ₁₀	1 0 1 0	0	m ₁₂ 1110
m ₁₁	1 0 1 1	0	m ₁₄ 1110
m ₁₂	1 1 0 0	-	m ₁₅ 1111
m ₁₃	1 1 0 1	0	
m ₁₄	1 1 1 0	1	
m ₁₅	1 1 1 1	-	

I₀ I₁ I₂ I₃ I₄ I₅ I₆ I₇ I₈ I₉ I₁₀

m₃ |

m₄ | | |

m₈ | | |

m₉ | | |

m₁₄ | | |

I₀ I₃ I₄ I₆ I₁₀

m₃

$$O = -100 + 100 - + 0 - 11 + 111 -$$

$$O = \overline{BC}\overline{D} + A\overline{B}\overline{C} + \overline{A}CD + A\overline{BC}$$

m₆

1

m₈

1

m₉

1

m₁₂

1

Es 2 (OH. Reti Combinatorie)

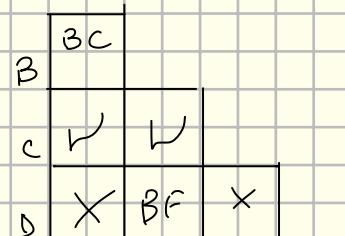
	a	b	c	d	e	O
m ₀	0	0	0	0	0	0
m ₁	0	0	0	0	1	0
m ₂	0	0	0	1	0	0
m ₃	0	0	0	1	1	0
m ₄	0	0	1	0	0	1
m ₅	0	0	1	0	1	0
m ₆	0	0	1	1	0	0
m ₇	0	0	1	1	1	0
m ₈	0	1	0	0	0	1
m ₉	0	1	0	0	1	0
m ₁₀	0	1	0	1	0	0
m ₁₁	0	1	0	1	1	0
m ₁₂	0	1	1	0	0	1
m ₁₃	0	1	1	0	1	0
m ₁₄	0	1	1	1	0	1
m ₁₅	0	1	1	1	1	0
m ₁₆	1	0	0	0	0	1
m ₁₇	1	0	0	0	1	0
m ₁₈	1	0	0	1	0	0
m ₁₉	1	0	0	1	1	0
m ₂₀	1	0	1	0	0	0
m ₂₁	1	0	1	0	1	-
m ₂₂	1	0	1	1	0	0
m ₂₃	1	0	1	1	1	0
m ₂₄	1	1	0	0	0	0
m ₂₅	1	1	0	0	1	0
m ₂₆	1	1	0	1	0	1
m ₂₇	1	1	0	1	1	0
m ₂₈	1	1	1	0	0	0
m ₂₉	1	1	1	0	1	0
m ₃₀	1	1	1	1	0	0
m ₃₁	1	1	1	1	1	1

$$\begin{array}{l}
 m_4 \quad 00100 \quad \text{m}_4 \quad 00100 \quad m_4 \quad m_{12} \quad 0-100 \quad I_4 \\
 m_8 \quad 01000 \quad m_8 \quad 01000 \quad m_8 \quad m_5 \quad 0100- \quad \checkmark \\
 m_{16} \quad 10000 \quad m_{16} \quad 10000 \quad m_8 \quad m_{12} \quad 01-00 \quad \checkmark \\
 \downarrow \\
 m_9 \quad 01001 \quad m_9 \quad 01001 \quad m_5 \quad m_{13} \quad 01-01 \quad \checkmark \\
 m_{12} \quad 01100 \quad m_{12} \quad 01100 \quad m_{12} \quad m_{14} \quad 011-0 \quad I_5 \\
 \downarrow \\
 m_{13} \quad 01101 \quad m_{13} \quad 01101 \quad m_{12} \quad m_{13} \quad 0110- \quad \checkmark \\
 m_{14} \quad 01110 \quad m_{14} \quad 01110 \quad \downarrow \\
 m_{21} \quad 10101 \quad m_{21} \quad 10101 \quad m_8 \quad m_5 \quad m_{12} \quad m_{13} \quad 01-0-\bar{I}_6 \\
 m_{26} \quad 11010 \quad m_{26} \quad 11010 \quad \downarrow \\
 \downarrow \\
 m_{31} \quad 11111 \quad m_{31} \quad 11111 \quad I_3 \\
 \downarrow \\
 I_0 \quad I_1 \quad I_2 \quad I_3 \quad I_4 \quad I_5 \quad I_6
 \end{array}$$

$$O = I_0 + I_1 + I_2 + I_3 + I_4 + I_5 + I_6$$

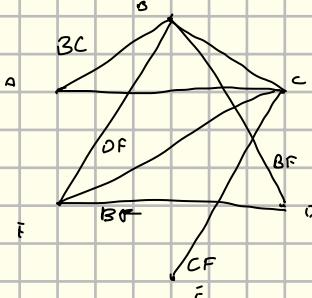
Es 1 (Optimizzazione Circuito Sequenziale)

	00	01	11	10
A	-/-	E/1	A/1-	C/1
B	F/0	-/-	-/-	B/1



C	-/-	-/-	C/0	-/-
D	-/-	B/-	E/1	F/1
E	A/-	E/0	F/-	-/-
F	D/0	F/-	-/-	-/-

E	X	X	CF	X
F	X	DF	V	BF



$$\{A, B, C\} = \alpha$$

$$\{F\} = \beta$$

$$\{D\} = \gamma$$

$$\{E\} = \mu$$

	00	01	11	10
α	-/-	M/1	$\alpha/-$	$\alpha/1$
β	$\gamma/0$	$\beta/-$	-/-	-/-
γ	-/-	$\omega/1$	$\mu/1$	$\beta/1$
μ	$\alpha/-$	$\mu/0$	$\beta/-$	-/-

Es 2 (Ottimizzazione Macchine Sequenti)

	00	01	11	10
A	-/-	$E/1$	$A/-$	$C/1$
B	-/-	$B/-$	$E/1$	$F/1$
C	$F/0$	-/-	-/-	$B/1$
D	$A/-$	-/-	$C/0$	-/-
E	$A/-$	$E/0$	$F/-$	-/-
F	$D/0$	$F/-$	-/-	$B/-$

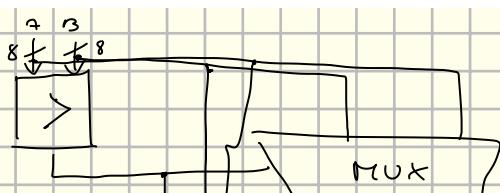
B	X
C	X
D	X
E	X
F	X

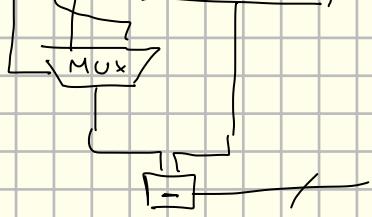
La macchina è già minima

Es 1 (Controllore Datapath)

Completere il datapath allegato, usando solo componenti di libreria, in modo che realizzzi la seguente funzione:

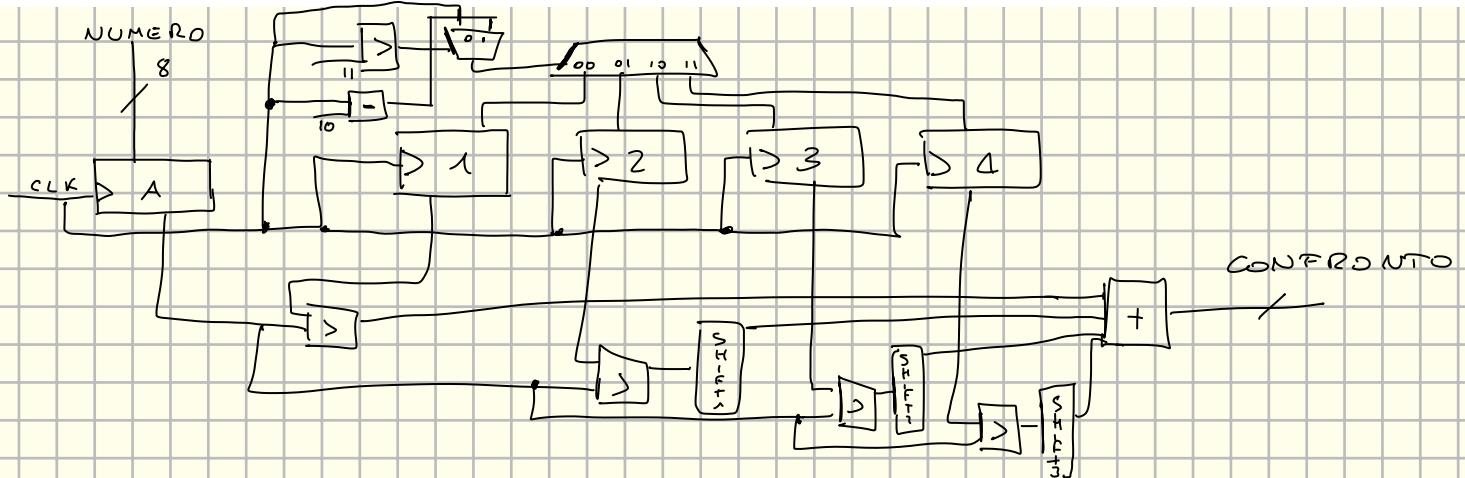
il circuito riceve in ingresso due numeri di 8 bit in modulo (A[8] e B[8]). Sottrae il numero minore dal maggiore, e fa apparire il risultato della sottrazione, un bit alla volta, sull'uscita RES[1] nei successivi 8 cicli di clock.





Es 2 (Controllore Datapath)

Utilizzando componenti di libreria si costruisca un datapath che legge, ad ogni ciclo di clock, un numero intero dall'ingresso NUMERO[8] e, confrontandolo con i 4 numeri precedentemente letti, pone, nello stesso ciclo di clock, sull'uscita CONFRONTO[4] un valore per ognuno dei confronti fatti. Questo valore vale 1 se il numero letto è maggiore, 0 se è minore o uguale.



Es 1 (Microistruzioni)

Fetch

- 1) PC_{out}, MAR_{IN}, READ, SELECT₄, ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}, END

Es 2 (Microistruzioni)

INC %EAX

- 1) PC_{out}, MAR_{IN}, READ, SELECT₄, ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}
- 4) EAX_{out}, SELECT₀, CB, ADD, Z_{IN}
- 5) Z_{out}, EAX_{IN}, END

Es 3 (Microistruzioni)

- 1) PC_{out}, MAR_{IN}, READ, SELECT₄, ADD, Z_{IN}
- 2) WMFC, Z_{out}, PC_{IN}
- 3) MDR_{out}, IR_{IN}
- 4) IR_{-imm-field out}, MAR_{IN}, READ, WMFC

- 5) MDR_{out}, SELECT₀, CB, ADD, Z_{in}
6) Z_{out}, MDR_{in}, WRITE, WMFC, END

Es 4 (Microistruzioni)

CALL ETICHETTA

- 1) PC_{out}, MAR_{in}, READ, SELECT₄, ADD, V_{in}
- 2) WMFC, Z_{out}, PC_{in}
- 3) MDR_{out}, IR_{in}
- 4) ESP_{out}, SELECT₄, SUB, Z_{in}
- 5) Z_{out}, ESP_{in}, MAR_{in}
- 6) PC_{out}, MDR_{in}, WRITE
- 7) IR-imm-field_{out}, MAR_{in}, READ
- 8) WMFC, PC_{out}, V_{in}
- 9) MDR_{out}, SELECT₀, ADD, Z_{in}
- 10) Z_{out}, PC_{in}, END

Es 5 (Microistruzioni)

CALL (%EAX, %EBX)

- 1) PC_{out}, MAR_{in}, READ, SELECT₄, ADD, Z_{in}
- 2) WMFC, Z_{out}, PC_{in}
- 3) MDR_{out}, IR_{in}
- 4) ESP_{out}, SELECT₄, SUB, Z_{in}
- 5) Z_{out}, MDR_{in}, ESP_{in}
- 6) PC_{out}, MDR_{in}, WMFC
- 7) WMFC, EAX_{out}, V_{in}
- 8) EBX_{out}, SELECT₀, ADD, Z_{in}
- 9) Z_{out}, MAR_{in}, READ
- 10) WMFC
- 11) MDR_{out}, PC_{in}, END

Es 6 (Microistruzioni)

RET

- 1) PC_{OUT}, MAP_{IN}, READ, SELECT_A, ADD, Z_{IN}
- 2) WMFC, Z_{OUT}, PC_{IN}
- 3) MDR_{OUT}, IR_{IN}
- 4) ESP_{OUT}, MAP_{IN}, READ, SELECT_C, ADD, Z_{IN}
- 5) WMFC, Z_{OUT}, ESP_{IN}
- 6) MDR_{OUT}, PC_{IN}, END

ES 7 (Microistruzioni)

JMP (%EAX)

- 1) PC_{OUT}, MAP_{IN}, READ, SELECT_C, ADD, Z_N
- 2) WMFC, Z_{OUT}, PC_{IN}
- 3) MDR_{OUT}, IR_{IN}
- 4) EAX_{OUT}, MAP_{IN}, READ
- 5) WMFC, PC_{OUT}, V_{IN}
- 6) MDR_{OUT}, SELECT_V, ADD, Z_{IN}
- 7) Z_{OUT}, PC_{IN}, END

ES 8 (Microistruzioni) JZ (%EAX)

- 1) PC_{OUT}, MAP_{IN}, READ, SELECT_A, ADD, Z_{IN}
- 2) WMFC, Z_{OUT}, PC_{IN}
- 3) MDR_{OUT}, IR_{IN}
- 4) if(!ZERO) END, EAX_{OUT}, MAP_{IN}, READ
- 5) WMFC, PC_{OUT}, V_{IN}
- 6) MDR_{OUT}, SELECT_V, ADD, Z_{IN}
- 7) Z_{OUT}, PC_{IN}, END

ES 9 (Microistruzioni) CALL (%EAX, %EBX)

- 1) PC_{OUT}, MAP_{IN}, READ, SELECT_A, ADD, Z_{IN}
- 2) WMFC, Z_{OUT}, PC_{IN}
- 3) MDR_{OUT}, IR_{IN}
- 4) ESP_{OUT}, SELECT_A, SUB, Z_N
- 5) -

5) ZOUT, MDR.in, ESP.in

6) PCout, MDR.in, WRITE

7) WMFC, EAX.out, JIN

8) EBX.out, SELECT4, ADD, Z.in

9) Zout, PC.in, END

ES 10 (Microistruzioni) CALL (%EAX, %EBX) Salto assoluto

1) PCout, MDR.in, READ, SELECT4, ADD, Z.in

2) WMFC, Zout, PC.in

3) MDR.out, IR.in

4) ESP.out, SELECT4, SUB, Z.in

5) Zout, ESP.in, MDR.in

6) PC.out, MDR.in, WRITE

7) WMFC, EAX.out, JIN

8) EBX.out, SELECT4, ADD, Z.in

9) Zout, MDR.in, READ

10) WMFC

11) MDR.out, PC.in, END

ES 11 (Microistruzioni) RET

1) PCout, MDR.in, READ, SELECT4, ADD, Z.in

2) WMFC, Zout, PC.in

3) MDR.out, IR.in

4) ESP.out, MDR.in, READ, SELECT4, ADD, Z.in

5) WMFC, Zout, ESP.in

6) MDR.out, PC.in, END

ES 12 (Microistruzioni) JMP (%EAX)

1) PCout, MDR.in, READ, SELECT4, Z.in

2) WMFC, Zout, PC.in

3) MDR.out, IR.in

4) EAX.out, MDR.in, READ

5) MDR.out, PC.in, END

ES 13 (Microistruzioni) CALL (% EAX, % EBX) assoluto

- 1) PCout, MARin, READ, SELECT_a, ADD, Zin
- 2) WIFC, Zout, PCin
- 3) MDRout, PCin
- 4) ESPout, SELECT_a, SUB, Zin
- 5) Zout, MARin, ESPin
- 6) PCout, MDRin, WRITE
- 7) WIFC, EAXout, V_in
- 8) EBXout, SELECT_v, ADD, Zin
- 9) Zout, MARin, READ
- 10) WIFC
- 11) MDRout, PCin, END

ES 14 (Microistruzioni) XCHG variabile, %eax

- 1) PCout, MARin, READ, SELECT_a, ADD, Zin
- 2) WIFC, Zout, PCin
- 3) MDRout, IRin
- 4) IR.imm_fieldout, MARin, READ
- 5) WIFC, EAXout, TEMPin
- 6) MDRout, EAXin
- 7) TMDout, MDRin, WRITE
- 8) WIFC
- 10) END

Es 1 (CACHE)

Esercizio 1 Prima parte

Si consideri una CPU dotata di memoria cache 4-associativa di 8K parole con 64 parole per blocco. Questa CPU è collegata ad una memoria RAM da 8M parole.

Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO ed ETICHETTA in cui questo indirizzo può essere suddiviso. Motivare la risposta con un opportuno schema.

CACHE 4-set
dim. CACHE 8K
parole 64
RAM = 8M

dim. indirizzo Mem. - $\log_2 2^3 \cdot 2^6 = 2^{13} = 8192$ bit

$$\text{parole} = \log_2 64 = \log_2 2^6 = 6 \text{ bit}$$

$$\text{blocco} = \log_2 \frac{2^3 \cdot 2^{10}}{2^6 \cdot 2^2} = \frac{2^{13}}{2^8} = 2^5 = 32 \text{ bit}$$

$$\text{etichette} = 23 - 5 - 6 = 12 \text{ bit}$$

etichetta : 12 bit	blocco: 5 bit	parola: 6 bit
--------------------	---------------	---------------

Esercizio 1 Seconda parte

Si assume che la cache appena descritta sia utilizzata per i dati, che sia inizialmente vuota e che utilizzi un algoritmo di sostituzione dei blocchi di tipo LRU (sostituzione dell'elemento meno utilizzato di recente). La CPU esegue un programma che accede in sequenza a tutti gli elementi di un array di 8320 parole (ogni elemento ha le dimensioni di una parola) che è memorizzato a partire dall'indirizzo 0. Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 5 volte. Si assume che il tempo di accesso alla cache sia di 1T e che il tempo di accesso alla memoria sia di 10T (entrambi i tempi si riferiscono alla lettura di una parola). Calcolare il rapporto (fattore di miglioramento) tra il sistema in presenza di cache e in assenza di cache per l'esecuzione di questo programma.

$$\text{alg} = \text{LRU}$$

$$\text{dim. Array} = 8320$$

$$\text{dim. Cache} = 2^{13} = 8192$$

$$T_c = 1 \quad T_{\text{RAM}} = 10 \quad \text{ciclo 5 volte}$$

$$T_{\text{senza cache}} = 8320 \cdot 10 \cdot 5 = 416000$$

$$\frac{8320}{64} = 130 \text{ blocchi} \quad \frac{8192}{64} = 128 \text{ blocchi}$$

$$1^\circ \text{ Set} = 130 \text{ Miss}$$

$$2^\circ - 5^\circ = (2 \cdot 4) + 2 = 10 \cdot 4 = 40 \text{ Miss}$$

$$\text{Miss Totali} = 170 \text{ Miss}$$

$$\text{Blocchi tot letti da RAM} = 170$$

$$\text{Parole lette da RAM} = 170 \cdot 64 = 10880$$

$$\text{Parole lette da CACHE} = (130 \cdot 64 \cdot 5) - 10880 = 30720$$

$$T \text{ lettura con CACHE} = (10880 \cdot 10) + 30720 \cdot 1 = 139520$$

$$\text{Speed up} = \frac{416000}{139520} = 2,98$$

Es 2 (CACHE)

Un programma consiste in un totale di 300 istruzioni e contiene un ciclo di 50 istruzioni che è eseguito 15 volte. Il processore contiene una cache. Prelevare ed eseguire un'istruzione che si trova in memoria principale richiede 20 unità di tempo. Se l'istruzione si trova nella cache, prelevare ed eseguire l'istruzione richiede solo 2 unità di tempo. Ignorando gli accessi dovuti al prelievo degli operandi, calcolare il rapporto tra il tempo di esecuzione del programma senza la cache e il tempo di esecuzione con la cache. Questo rapporto è chiamato accelerazione (speedup) dovuta all'uso della cache. Assumere che la cache sia inizialmente vuota, che sia sufficientemente grande da contenere il ciclo e che il programma inizi con tutte le istruzioni presenti in memoria principale.

$$1\text{str} = 300 \quad 50 \text{ istri} \rightarrow \text{ciclo} = 15$$

$$T_{mem} = 20T \quad T_c = 2T$$

$$T_c 250 \cdot 20 + 50 \cdot 20 \cdot 15 = 20000$$

$$T_{cc} = 300 \cdot 20 + 50 \cdot 2 \cdot 14 = 7400$$

$$\frac{T_c}{T_{cc}} = 2,7$$

ES 3 (CACHE)

Cache 2-set

dim. Cache = 64 K

parole = 1024

RAM = 4 M

dim. parte indirizzabile RAM = $\log_2 2^2 \cdot 2^{20} = \log_2 2^{22} = 22$ bit

parola = $1024 = 2^{10} \rightarrow \log_2 2^{10} = 10$ bit

blocco = $\log_2 \frac{2^6 \cdot 2^{10}}{2^{10} \cdot 2^1} = \frac{2^{16}}{2^{11}} = 5$ bit

etichette = $22 - 10 - 5 = 7$ bit

ES 4 (CACHE)

Cache 2-set dim. Cache = 8K

parole = 64 RAM = 8M

indirizzo RAM = $2^3 \cdot 2^{20} \rightarrow \log_2 2^{23} = 23$ bit

blocco $\rightarrow \log_2 \frac{\text{dim Cache}}{\text{parole} \cdot \text{set}} = \frac{2^{13}}{2^6 \cdot 2^1} = 6$ bit

parole $\rightarrow \log_2 2^6 = 6$ bit

etichette $\rightarrow 23 - 6 - 6 = 11$ bit

11 bit	6 bit	6 bit
etichette	blocco	parola

alg = LRU

dim. Array = 8320

Tc = 1T

Esercizio 3

Si assuma che un computer abbia memoria virtuale strutturata a pagine di 4Kbyte.

Si consideri un programma con un codice di 7.2 Kbyte che accede ciclicamente in sequenza a tutti gli elementi di un array di 1000 record in cui ogni campo è composto da 2 numeri interi ciascuno di 4 byte.

Quale deve essere la dimensione del *working set* perché si abbiano dei *page fault* solamente nella fase di caricamento del programma con esecuzione del primo ciclo di accesso agli elementi dell'array?

Quanti *page fault* si avrebbero durante l'esecuzione del programma ipotizzando che il *working set* abbia una pagina meno di quanto definito nel punto precedente e che il ciclo di accesso venga ripetuto 10 volte?

$$\text{mem. Virtuale} = 4 \text{K} \rightarrow 2^2 \cdot 2^{10} = 2^{12} = 4096$$

$$\text{codice} \rightarrow 7.2 \text{K} \rightarrow 2 \text{ pagine}$$

$$\text{dim. array} = 1000 \cdot 2 \cdot 4 = 8000$$

$$\frac{8000}{4096} = 1,95 \rightarrow \text{servono 2 pagine}$$

a) Servono 4 pagine $\rightarrow 2$ per il codice e 2 per i dati

b) Con un ws di 3 pagine ed un ciclo di 10

1° ciclo \rightarrow 4 page fault

2°-9° ciclo \rightarrow 2·3 page fault

$$18 + 4 = 22 \text{ page fault}$$

Es 5 (Cache)

completamente associativa

$$\text{dim Cache} = 64 \text{ K}$$

$$1024 \text{ per blocco}$$

$$\text{RAM} = 16 \text{ M}$$

$$\text{indirizzo RAM} = \log_2 16 \text{M} = \log_2 2^4 \cdot 2^{20} = 2^{24} = 24 \text{ bit}$$

$$\text{parole} = 1024 \rightarrow 2^{10} \rightarrow \log_2 2^{10} = 10 \text{ bit}$$

$$\text{etichette} = 24 - 10 = 14 \text{ bit}$$

Es 1 (Pipeline)

istr 1 2 3 4 S 6 7 8 9 10 11

addl %eax, %ebx F D E M S

movl \$4, %ecx F D E M S

subl %ebx, %ecx F D D P D E M S

movl \$4, %edx F F F F D E M S

Es 3 (Pipeline)

istr 1 2 3 4 S 6 7 8 9 10 11

ciclo	addl %eax, %ebx	F D E M S	F D D E M
movl	\$4, %ecx	F D E M S	F F D M
subl	%eax, %edx	F D E M S	F D
movl	\$6, %ebx	F D E M S	F
jmp	ciclo'	F P E M S	

Es 4 (Pipeline)

istr 1 2 3 4 S 6 7 8 9 10 11 12 13

inizio: inc %ebx	F D E M S	F D E M S
movl %ecx, %edx	F D E M S	F D E M
cmpl %eax, 0x86FF	F D E M S	F D E
jne inizio #yes	F D D D D E M S F D	
movl %ecx, %edx	FF F F	F

Es 5 (Pipeline)

istc 1 2 3 4 S 6 7 8 9 10 11

START: subl %eax, %ebx	F D E M S
jz START #no	F D D D D E M S
subl %ebx, %ecx	FF F F D E M S
movl %edx, %eax	F D E M S

Es 6 (Pipeline)

istr 1 2 3 4 S 6 7 8 9 10 11 12 13 14 15

ciclo: addl %eax, %ebx	F D E M S
movl %edx, %ecx	F D E M S
subl %ebx, %ecx	F D D D D E M S
jz ciclo #m	FF F F D D D D E M S
movl %ecx, %edx	F F F F D E M S

Es 7 (Pipeline)

istr 1 2 3 4 S 6 7 8 9 10 11 12 13

ciclo addl %eax, %ebx	F D E W
movl %ebx, %ecx	F D D D E W
subl %eax, %ecx	FF F D D D E W
jz ciclo #m	FF F D D D E W

Es 8 (Pipeline)

istr 1 2 3 4 S 6 7 8 9 10 11 12 13 14 15

init movl %ecx, %edx	F D E M S
addl \$4, %ebx	F D E M S
cmpl 0x319FA, %ebx	F D D D D E M S
jne #no	F F F D D D D E M S

addl %eax, %ebx

F F F F D E M S

lstr

1 2 3 4 S 6 7 8 9 10 11 12 13 14 15 16 17 18

```

init moul %ecx, %edx
addl $4 %ebx
cmpl 0x319FA, %ebx
jnz # yes
addl %eax, %ecx

```

F D E M S	F D E M S
F D E M S	F D E M S
F D D D D E T S	F D D D D
F F F F D D D D E M S F F F F	F F F F

Esercizio (Pipeline)

lstr

1 2 3 4 S 6 7 8 9 10 11

```

loop addl %eax, %ebx
moul ind %ecx
subl %ebx, %ecx
jz loop

```

F D E W	
F D E W	
F D D D E W	
F F F D D D E W	

branch prediction
↓

lstr

1 2 3 4 S 6 7 8 9 10 11

```

loop addl %eax, %ebx
moul ind %ecx
subl %ebx, %ecx
jz loop

```

F D E W	F D E
F D E W	F D
F D D D E W	F
F F F D D D E W	

Esercizio (Pipeline)

lstr

1 2 3 4 S 6 7 8 9 10 11

```

inizio: mull %eax, %ebx
moul %ecx ind
cmpl %ebx, 0h
jnz inizio

```

F D E W	F
F D E W	
F D D E W	
F F D D D E W	

branch prediction
↓

lstr

1 2 3 4 S 6 7 8 9 10 11

```

inizio: mull %eax, %ebx
moul %ecx ind
cmpl %ebx, 0h
jnz inizio

```

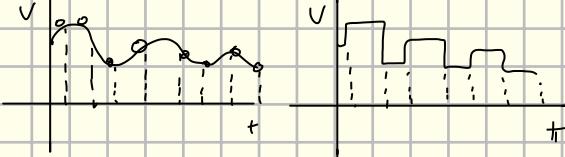
F D E W	F D E W
F D E W	F D E
F D D E W	F D
F F D D D E W	F

Esercizi Appelli

- 1) Descrivere (possibilmente attraverso un grafico) i concetti di discretizzazione e campionamento necessari alla trasformazione di una grandezza analogica continua nel tempo in una grandezza digitale.

Lo reale non è descritto da una serie di numeri. Per descrivere delle grandezze reali in un sistema digitale dobbiamo decidere un "errore" da compiere. Il teorema di Shannon ci dice

che deciso l'errore da compiere esiste una frequenza di campionamento ed un intervallo di discretizzazione che garantiscono quell'errore.



Nel primo grafico stiamo campionando la realtà con intervalli di tempo finiti. Nel secondo grafico discretizziamo i numeri reali, trasformandoli in razionali.

2) Sottrarre il numero binario 1101 dal numero binario 0101011, interpretando entrambi i numeri come codificati in complemento a due, ed esprimere il risultato come numero in base dieci. Quale risultato si sarebbe ottenuto se si fossero considerati i numeri codificati in modulo?

$$0101011 \rightarrow 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^3 + 1 \cdot 2^5 = 43_{10}$$

$$\begin{array}{r} 1101_2 \\ - 0101_2 \\ \hline 1100_2 \end{array} \rightarrow 0011_2 \rightarrow -3_{10}$$

$$43 - (-3) = 46_{10}$$

$$\begin{array}{r} 0101011 \\ + 0000011 \\ \hline 0101110_2 \end{array} \rightarrow 46_{10}$$

$$\text{MODULO} = 0101011_2 \rightarrow 43_{10} \quad (\text{upuole al CA2})$$

$$1101_2 \rightarrow 1 \cdot 2^0 + 1 \cdot 2^2 + 1 \cdot 2^3 = 13_{10}$$

$$43 - 13 = 30_{10}$$

$$\begin{array}{r} 0101011 \\ - 0001101 \\ \hline 0011110_2 \end{array} \rightarrow 30_{10}$$

4) Si definiscano, possibilmente aiutandosi con un esempio, i concetti di: implicante, implicante primo e implicante primo essenziale.

Implicante \rightarrow è un prodotto di letterali dello spazio Booleano d'insieme, tale che se il prodotto vale 1 allora la funzione vale 1.

Implicante primo \rightarrow è un implicante non coperto da nessun altro implicante di dimensioni maggiori.

Implicante essenziale \rightarrow è un implicante che contiene un minimo che non coperte da nessun altro implicante.

a	b	c	d	00	01	11	10
00	1			1			
01		1			1		
11			1				
10				1			

implicante essenziale
implicante primo
implicante

1) Dato il numero 1.16 in notazione decimale, convertirlo in virgola mobile, singola precisione.

Si utilizzi 4 bit per la parte intera e 5 bit per la parte decimale.

Commentare opportunatamente i passaggi effettuati.

$$1,16_{10} \rightarrow 0001,00101_2$$

$$\begin{array}{r} 0,16 \cdot 2 = 0 \\ 0,32 \cdot 2 = 0 \\ 0,64 \cdot 2 = 1 \\ 0,28 \cdot 2 = 0 \\ 0,56 \cdot 2 = 1 \end{array}$$

$$1,16_{10} > 0 \rightarrow \text{segno} = + \rightarrow \text{bit } 0$$

$$\text{esponente} = 0 + 130 = 130$$

$$\begin{array}{r} 130/2 = 0 \\ GS/2 = 1 \\ 32/2 = 0 \\ 16/2 = 0 \\ 8/2 = 0 \\ 4/2 = 0 \\ 2/2 = 0 \\ 1/2 = 1 \\ 0 \end{array} \quad \text{esponente} = 1000001_0$$

$$\underbrace{01000001}_\text{segno} \underbrace{00101}_\text{esponente} \underbrace{000000000000000000000000}_\text{mantissa}$$

2) Data la funzione $f = (a, b, c, d)$ con $On-set = \{m_0, m_1, m_2, m_4, m_6\}$, eseguire l'algoritmo Quine – McCluskey.

Nel risultato, evidenziare il numero di implicant primi essenziali, scrivere il numero di letterali e scrivere la funzione ottimizzata tramite la somma di prodotti.

Esempio: Gli implicant primi essenziali sono 3: $A + C + E = \bar{a}b + c + \bar{a}cd$. I letterali sono in totale 4.

Commentare opportunatamente i passaggi effettuati.

ab	cd	0
0 0	0 0	1 m_0
0 0	0 1	1 m_1
0 0	1 0	1 m_2
0 0	1 1	0 m_3
0 1	0 0	1 m_4
0 1	0 1	0 m_5
0 1	1 0	1 m_6
0 1	1 1	0 m_7
1 0	0 0	0 m_8
1 0	0 1	0 m_9
1 0	1 0	0 m_{10}
1 0	1 1	0 m_{11}
1 1	0 0	0 m_{12}
1 1	0 1	0 m_{13}
1 1	1 0	0 m_{14}
1 1	1 1	0 m_{15}

mo	0000	→	$m_{0,1} 000 - I_0$	$m_{0,2} m_{4,6} 0 - 0 I_1$
m_1	0001		$m_{0,2} 00 - 0 V$	
m_2	0010		$m_{0,4} 0 - 00 V$	
m_4	0100		$m_{2,6} 0 - 10 V$	
m_6	0110		$m_{4,6} 01 - 0 V$	

$I_0 \quad I_1 \rightarrow$ entrambi essenziali

$$0 \rightarrow \bar{A}\bar{B}\bar{C} + \bar{A}\bar{D}$$

3) Data la seguente STT:

	0	1
A	A/0	A/0
B	A/0	F/1
C	B/1	B/0
D	A/1	C/0
E	F/0	E/0
F	A/0	F/0

Eseguire la minimizzazione ed evidenziare la/le classe/i individuata/e.

B	x			
C	x	x		
D	x	x	~	
E	~	x	x	x
F	~	x	x	x

$$\{A \in F\} = A$$

$$\{CD\} = C$$

$$B = B$$

A	0	1
A	A/0	A/0

B	A/0	A/1
---	-----	-----

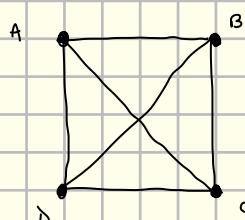
C	B/1	B/0
---	-----	-----

4) Data la seguente FSM:

	00	01	11	10
A	B/1	D/0	B/1	B/1
B	D/1	B/1	B/0	B/0
C	C/0	B/0	D/1	B/0
D	D/1	D/0	C/1	B/0

Eseguire la codifica ipotizzando che lo stato A abbia già una codifica di 00. In questo modo si limiteranno le opzioni di codifica.

Evidenziare la codifica ricavata.



$$\begin{aligned}
 AB &= 123 \\
 AC &= 123 \\
 AD &= 12345 \\
 BC &= 1234 \\
 BD &= 12345678 \\
 CD &= 12345
 \end{aligned}$$

$$\begin{aligned}
 BD &= 8 \\
 CD - AD &= 5 \\
 BC &= 4 \\
 AB - BC &= 3
 \end{aligned}$$

00	A
10	D
11	B
01	C

2) Identificare le parti logiche dell'indirizzo RAM di una memoria da 256 MB, indirizzabile per byte, nel caso sia collegata ad una memoria cache associativa a gruppi, con 4 posizioni per gruppo, da 32KB in cui ogni posizione immagazzina un blocco di memoria di 16 byte.

- Descrivere i passaggi coinvolti nella conversione di un indirizzo logico in uno fisico, assumendo la presenza di TLB. Spiegare le conseguenze di un "hit" o "miss" in ogni consultazione di tabella.

Cache 4-set

RAM = 256 MB

CACHE = 32 kB

blocco = 16

$$\text{bit. indirizzabilità RAM} = \log_2 2^8 \cdot 2^{20} = 2^{28} = 28 \text{ bit}$$

$$\text{parole} = \log_2 \frac{16}{2^3 \cdot 2^{10}} = 2^4 = 4 \text{ bit}$$

$$\text{blocco} = \log_2 \frac{2^3}{2^4 \cdot 2^2} = \log_2 \frac{2^3}{2^6} = 2^3 = 3 \text{ bit}$$

$$\text{etichette} = 28 - 3 - 4 = 15 \text{ bit}$$

15 bit	3 bit	4 bit
etichetta	blocco/set	parole

• TLB \rightarrow Cache d: traduzione indirizzi

Il processore genera un indirizzo logico e consulta lo TLB

TLB-HIT: se la pagina è presente nelle TLB si ottiene immediatamente la pagina risce corrispondente

TLB-MISS: se non è presente lo si recupera poi caricarla

3) Elenicare e commentare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un BUS, che l'istruzione sia composta da una sola parola, che \$xx(%Exx) rappresenti un metodo di indirizzamento indiretto a registro con spiazzamento e che l'indirizzo del salto sia relativo (usare solamente le righe necessarie e commentare ogni istruzione):

CALL \$4(%EAX)

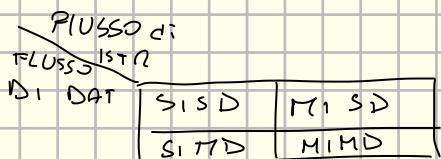
commento

- 1) PCout, MAin, READ, SELECT, ADD, Zin
- 2) WMFC, Zout, PCin
- 3) MD2out, IRin
- 4) EAXout, MA2in, READ
- 5) WMFC
- 6) MDout, Vin
- 7) IRout, SELECTv, ADD, Zin
- 8) Zout, Vin
- 9) PCout, SELECTv, ADD, Zin
- 10) Zout, PCin, END

4) Riportare la Tassonomia di Flynn sulle architetture parallele e dire a quali realizzazioni odierne corrispondono le tre architetture possibili.

- Discutere l'impatto che hanno su una architettura RISC o CISC le tre parti della formula che descrive il tempo di esecuzione (T_{cpu}) di un programma composto da N istruzioni eseguite alla frequenza f .

Le Tassonomia di Flynn parla del concetto di realizzazione di architetture parallele. Avendo quindi flussi paralleli di dati o flussi paralleli di istruzioni.



SISD → architettura di Von Neumann dove il processore esegue un istr. alla volta

SIMD → schede grafiche che eseguono una singola istr. su più dati

MIMD → NOC evolute poi nelle CPU multicore

Si hanno + CPU ma il Bus c'è sempre saturo.
C'è un limite oltre al quale non ci sono + miglioramento.
(4-8).

RISC - CISC :

$$\text{La formula } T_{cpu} = \# \text{ istr.} \cdot CPI_m \cdot \frac{1}{f_{ck}}$$

per migliorare il tempo bisognerebbe diminuire le istr. o aumentare la frequenza.

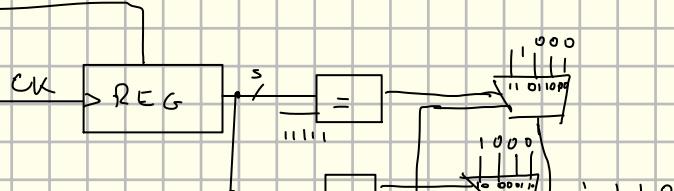
RISC → Reduced → aumenta il n° di istr. di istruzioni si avrebbe un aumento della frequenza

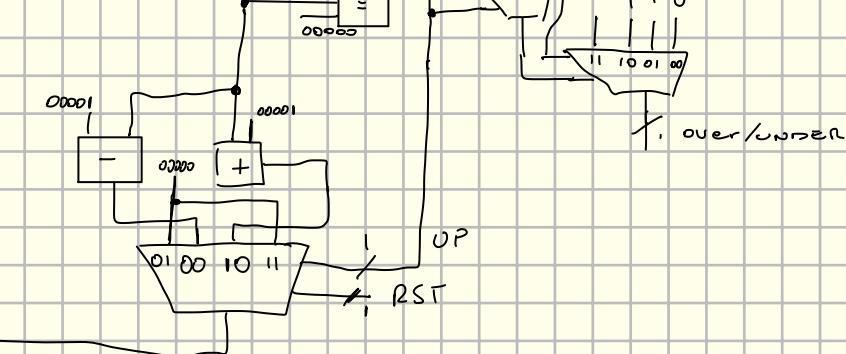
CISC → Complex → diminuisce il n° di istr. e diminuisce la frequenza

- 1) Si realizzi un Datapath in grado di effettuare il conteggio verso l'alto oppure il conteggio verso il basso, in base al valore dell'ingresso ad 1 bit chiamato UP. Si assuma che il valore del conteggio sia memorizzato in 5 bit. L'ingresso RST resetta il conteggio a 0 quando RST=1, mentre l'uscita OVER/UNDER va posta a 1 se il conteggio vale 0 a causa di un overflow o un underflow. Si dispone in libreria dei blocchi elementari sommatore, sottrattore, controllo di uguaglianza, multiplexer e registro.

INPUT → UP[1] RST[1] → 1 resetta il collettore

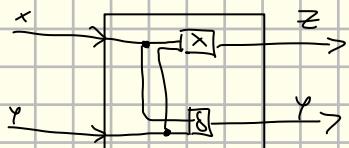
OUTPUT [1] → 1 se il conteggio vale 0 per over/under



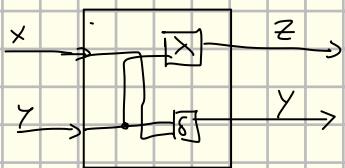


- Disegnare il modello di Huffman nel caso di macchina di Mealy e di Moore

Mealy \rightarrow uscita in funzione dello stato d'ingresso



Moore \rightarrow uscita in funzione dello stato



- 3) Elencare e commentare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un BUS, che l'istruzione sia composta da una sola parola, che \$xx(%Exx) rappresenti un metodo di indirizzamento indiretto a registro con spiazzamento e che l'indirizzo del salto sia assoluto (usare solamente le righe necessarie e commentare ogni istruzione):

JZ \$8(%EAX)

- 1) PCout, MARin, READ, SELECT+, ADD, Z,N
- 2) WIFC, Zout, PCin
- 3) MDout, IRin
- 4) IF (!ZERO) END EAXout, MARin, READ
- 5) WMFC, Rout, Vin
- 6) MDout, SELECT+, ADD, Z,N
- 7) Zout, PCin, END

- 2) Si consideri una cache con hit rate per le istruzioni 90% ed hit rate per i dati 80%. Il tempo di accesso alla cache è 1 ciclo, il tempo di accesso alla memoria è 18 cicli. Si consideri un programma in cui il 20% delle istruzioni è composto di accessi a memoria. Si ipotizza per semplicità che sia per miss di lettura che per miss di scrittura si perda 1 ciclo extra per il secondo accesso alla cache (oltre al tempo di accesso a memoria). Si calcoli lo speedup dovuto alla presenza di tale cache rispetto ad una architettura senza cache.

Senza CACHE

Fetch \rightarrow 18 cicli

Accesso ai dati 18 cicli (20%)

$$18 + 0,20 \cdot 18 = 18 + 3,6 = 21,6 \text{ cicli/istruzione}$$

Con CACHE

Miss costa 20 cicli

Dati

Con probabilità 90% = 1 ciclo
10% = 20 cicli

$$0,90 \cdot 1 + 0,10 \cdot 20 = 2,9 \text{ cicli/istruzione}$$

Istr.

Con probabilità 80% = 1 ciclo
con probabilità 20% = 20 cicli

$$0,80 \cdot 1 + 0,20 \cdot 20 = 4,8$$

quindi per il 80% avremo solo 2,9 cicli/istruzione

20% = 7,7 cicli

$$\text{Tutto} = (0,8 \cdot 2,9) + (0,2 \cdot 7,7) = 3,86$$

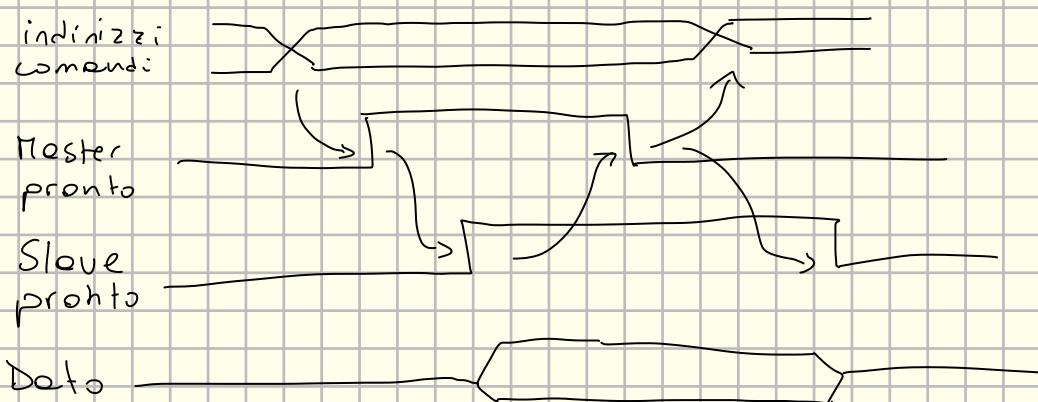
$$\frac{21,6}{3,86} = 5,6$$

• Come funziona l'indirizzamento indiretto nell'LC3?

L'indirizzamento indiretto nell'LC3 funziona utilizzando la somma PC + Offset (12). Questo risultato esce dall'ALU e viene utilizzato come indirizzo alle celle di memoria dove trovare ciò che cerchiamo

- Disegnare il diagramma temporale dell'evoluzione dei segnali di un BUS asincrono per realizzare la lettura di un dato da Memoria sotto il controllo della CPU.

esempio di lettura:



indirizzi e comandi variano perché il master deve fare una lettura

Master pronto si alza ad uno perché inizia una nuova operazione

Slave pronto si alza come risposta al master

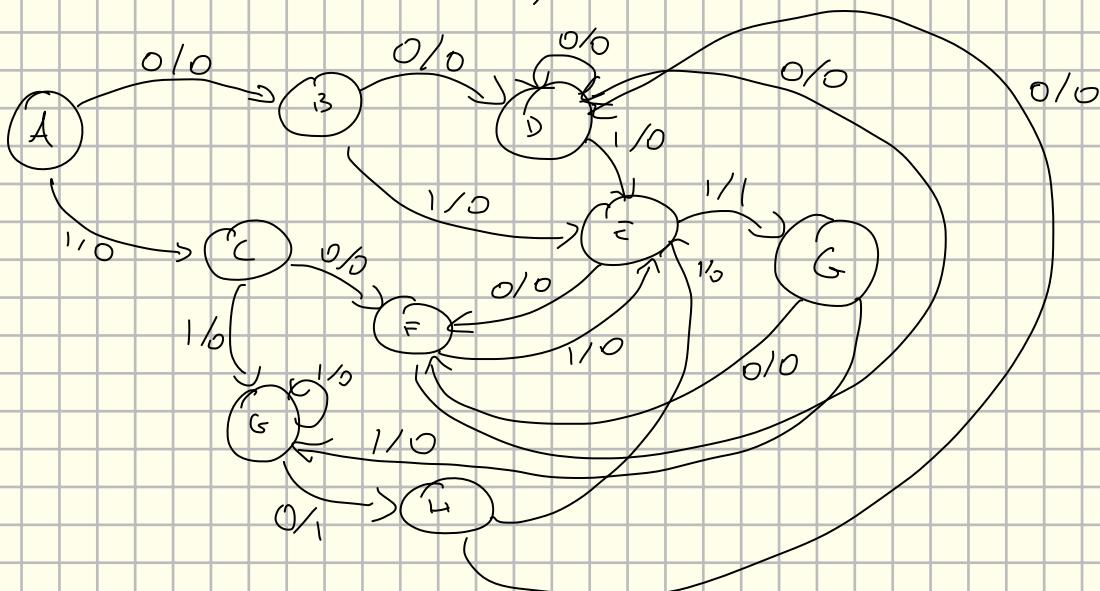
Quando lo slave è pronto master si abbassa, il dato rimane valido nel tempo di slave pronto

Questi bus asincroni: come visto funzionano utilizzando i segnali (Handshaking).

- 1) Si disegni il grafo di transizione di stato per la FSM di Mealy che implementa la seguente specifica: dato un ingresso X ad 1 bit, l'uscita Z ad 1 bit vale 1 quando si presentano le sequenze (denominate " valide") di ingresso 010 oppure 110, altrimenti Z vale 0; le sequenze valide possono sovrapporsi.

X[1]

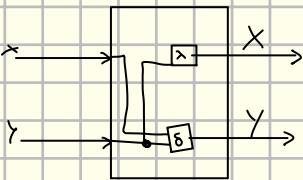
Z[1] \rightarrow 1 (010) (110)



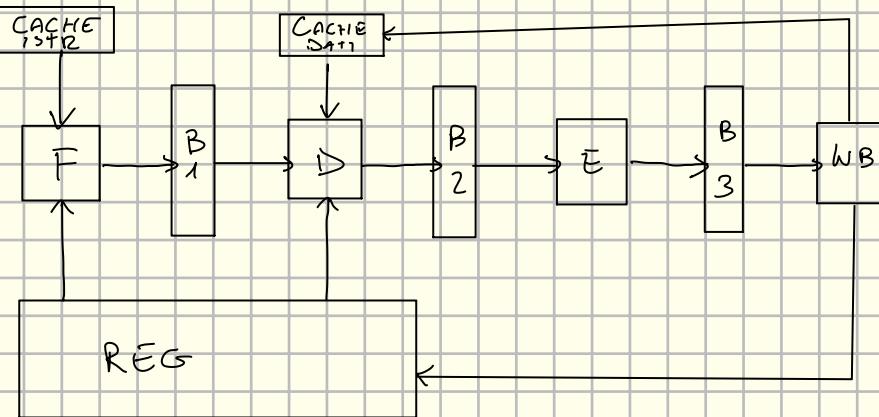
0 1

A	B/0	C/0
B	D/0	E/0
C	F/0	G/0
D	D/0	E/0
E	F/0	G/1
F	D/0	E/0
G	H/1	G/0
H	D/0	E/0

- Modello Huffman per file
- Uscite in funzione degli stati



1.1) Si disegni lo schema di un'architettura a pipeline a 4 stadi e si spieghi come sia possibile che il CPI medio possa tendere a 1. Quali sono le motivazioni che tendono a produrre un CPI medio superiore a 1?



Con le pipeline svolgeremo più istruzioni in meno cicli macchine
aggiungendo altre migliorie come cache istr, cache dati, set associative...
si ridurranno ancora di più il CPIm.
Ciò però fa fronte a diversi problemi: come le dipendenze,
limiti di risorse e ritardi fra le memorie.

1.2) Qual è l'espressione a 3 fattori che descrive il tempo di esecuzione di un programma su una CPU? Quali di questi 3 fattori sono migliori in un'architettura CISC e quali in una RISC?

$$T_{CPU} = \# \text{istr} \cdot CPI_m \cdot \frac{1}{f_{clk}}$$

nell' architettura

CISC \rightarrow abbiamo un numero minore di istruzioni che sono però + complesse
Facendo aumentare le f di clock

RISC \rightarrow aumentiamo (rendendole + semplici) le istruzioni e diminuiamo le f del clock

2.1) Si consideri una cache di 4KB con indirizzamento diretto, in cui ogni posizione immagazzina 8 parole di memoria. La memoria è di 32MB, con parole di 8bit. A quanti bit corrisponde il campo etichetta di una data posizione della cache?

dim. Cache = 4 KB

dim. item = 32 MB \rightarrow parole d: 8 bit

dim. Blocco = 8 B

$$\text{indirizzamento RAM} = \log_2 2^6 \cdot 2^{20} = 2^{25} \text{ bit}$$

$$\text{blocco} = \log_2 \frac{\text{dim Cache}}{\text{dim Bloco}} = \log_2 \frac{2^6 \cdot 2^{20}}{2^3} = 2^8 = 9 \text{ bit}$$

$$\text{etichette} = 25 - 9 - 8 = 8 \text{ bit}$$

3) Si consideri un sistema a memoria virtuale con spazio logico di 4G parole, una memoria fisica di 32M parole e dimensione delle pagine di 16K parole.

- Determinare il numero di bit che definiscono:

Lunghezza indirizzo logico:

Di cui per Num. Pag. Logica

per lo spiazzamento

Lunghezza indirizzo fisico

di cui per Num. Pag. Fisica

per lo spiazzamento

$$\text{dim. ind. logico} = \log_2 2^7 \cdot 2^{30} = 32 \text{ bit}$$

$$\text{dim. pagina logico} = \log_2 \frac{2^{32}}{2^{10}} = 2^{18} = 18 \text{ bit}$$

$$\text{dim. ind. fisico} = \log_2 2^5 \cdot 2^{20} = 2^{25} = 25 \text{ bit}$$

$$\text{dim. pagina fisico} = \log_2 \frac{2^{23}}{2^{12}} = 11 \text{ bit}$$

$$\text{offset} = \log_2 2^4 \cdot 2^{10} = 14 \text{ bit}$$

Esercizio 3 Seconda parte

- Nella seguente tabella sono riportati alcuni valori del parametro R (numero di pagine residenti per processo); sapendo che il Sistema Operativo occupa permanentemente 448 pagine e che sono stati creati 22 processi, indicare per ogni valore di R il numero di processi in stato di "fuori memoria" ossia che non possono avere tutte le pagine in memoria.

Dato **R** il numero di pagine residenti per processo, si ha:

Sistema Operativo occupa 448 pagine permanentemente

Numero Processi 22

$$\text{n° pagine totali} = 2^9 = 2048$$

$$2048 - 448 = 1600 \text{ pagine libere}$$

R	40	80	160	320	800
Nº processi fuori memoria	0	2	12	17	20

$$\frac{1600}{40} = 40$$

$$\frac{1600}{80} = 20$$

$$\frac{1600}{160} = 10$$

$$\frac{1600}{320} = 5$$

↓

$$40 > 22 = 0$$

$$22 - 20 = 2$$

$$22 - 10 = 12$$

processi fuori mem

Esercizio 4

Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia relativo al PC:

CALL (%EAX)

- 1) PC_{out}, MAR_{in}, READ, SELECT₄, ADD, Z_{IN}
- 2) WIFC, Z_{out}, PC_{IN}
- 3) MDROUT, IR_{IN}
- 4) ESPOUT, SELECT₄, SUB, Z_{IN}
- 5) Z_{out}, ESPIN, MAR_{IN}
- 6) PC_{out}, MDR_{IN}, WRITE
- 7) WIFC
- 8) GAX_{OUT}, MAR_{IN}, READ
- 9) WIFC, PC_{OUT}, V_{IN}
- 10) MDROUT, SELECT₄, ADD, Z_{IN}
- 11) Z_{out}, PC_{IN}, END

Esercizio 3 Prima parte

Si consideri una memoria cache 4-set associativa della dimensione di 32 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 1Mbyte indirizzabile per byte. Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

4 - Set

$$\begin{aligned} \text{dim. Cache} &= 32 \text{ KByte} \\ \text{dim. blocco} &= 1024 \text{ byte} \\ \text{dim. Mem.} &= 1 \text{ MB} \end{aligned}$$

$$\text{indirizzo delle cache} = \log_2 2^5 \cdot 2^{10} = 15 \text{ bit}$$

$$\text{indirizzo delle memorie} = \log_2 2^{20} = 20 \text{ bit}$$

RAM :

$$\text{parole} = 1^{10} = 10 \text{ bit}$$

$$\text{blocco/set} = \log_2 \frac{2^{20}}{2^{10} \cdot 2^2} = 8 \text{ bit}$$

$$\text{etichette} = 20 - 10 - 8 = 2 \text{ bit}$$

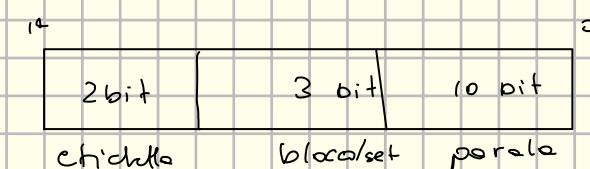


Cache :

$$\text{parole} = 10 \text{ bit}$$

$$\text{blocco/set} = \log_2 \frac{2^{15}}{2^{10} \cdot 2^2} = 3 \text{ bit}$$

$$\text{etichette} = 15 - 10 - 3 = 2 \text{ bit}$$



Esercizio 3

Si consideri un programma che accede in sequenza a tutti gli elementi di un array di 10000 parole (ogni elemento ha dimensione di una parola). Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 10 volte.

Si assuma che le dimensioni di una pagina siano di 1k parole. Il programma occupa due pagine ed il suo working set è composto da 10 pagine.

Quanti page fault avvengono durante l'esecuzione del programma considerando che nessuna pagina del programma è residente in memoria al momento della sua esecuzione?

dim. Array 10000 parole
ciclo 10 volte

pagine = 1024 parole

programma occupa 2 pagine

$$\frac{10000}{1024} = 9,7 \text{ servono 10 pagine}$$

1° ciclo 12 page fault

$2 - 9 = 9$ ° page fault

102 page fault totali

Esercizio 4

- Si consideri una CPU con una pipeline a 4 stadi (F, D, O, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

istr	1	2	3	4	S	6	7	8	9	10	11
null	%eax, %ebx				F	D	E	W			
movl	%eax, ind				F	D	E	W			
cmp	%ebx, 0h				F	D	D	E	W		
jnz	init10				F	F	D	D	D	E	W

Esercizio 3 Prima parte

Si consideri una memoria cache 4-set associativa della dimensione di 16 Kbyte con 512 byte per blocco. La cache è collegata ad una memoria di 2Mbyte indirizzabile per byte.

- Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

cache 4-set

$$\begin{aligned} \text{dim Cache} &= 16 \text{ KB} \\ \text{dim blocco} &= 512 \text{ byte} \end{aligned}$$

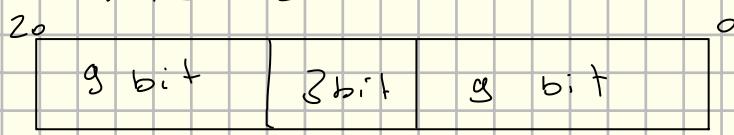
$$\text{dim RAM} = 2MB$$

$$\text{indirizzo RAM} = \log_2 2^{\text{20}} = 2^{\text{20}} = 2^{\text{21}} = 21 \text{ bit}$$

$$\text{parola} = \log_2 2 = 1 \text{ bit}$$

$$\text{blocco/}set = \log_2 \frac{2^{\text{10}}}{2^{\text{3}}} = \frac{2^{\text{10}}}{2^{\text{3}}} = 2^{\text{7}} = 128 \text{ byte}$$

$$\text{etichette} = 21 - 9 - 3 = 9 \text{ bit}$$

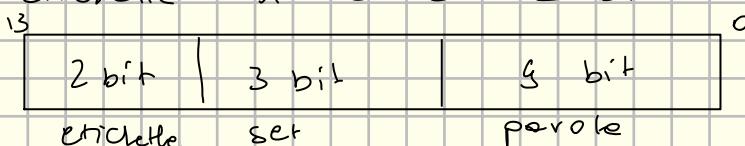


etichette set parole

$$\text{indirizzo Cache} = \log_2 2^{\text{4}} \cdot 2^{\text{10}} = 14 \text{ bit}$$

$$set = 3 \text{ bit} \quad \text{parole} = 5 \text{ bit}$$

$$\text{etichette} = 14 - 3 - 5 = 6 \text{ bit}$$



etichette set parole

Esercizio 1 Prima parte

Elencare le micro istruzioni relative al caricamento, decodifica ed esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che il salto condizionato sia di tipo diretto:
JZ (%EAX)

→ assoluto

- 1) PCout, MARin, 2EAD, SELECTa, ADD, Zin
- 2) WIFC, Zout, PCin
- 3) MDRout, IRin
- 4) IF (!Z=0) END, EAXout, MARin, 2EAD
- 5) WIFC
- 6) MDRout, PCin, END

Esercizio 2

Si consideri una CPU dotata di memoria cache di 128K byte con 64 byte per blocco. Questa CPU è collegata ad una memoria RAM da 16M byte indirizzabile per byte.

- Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO (o SET) ed ETICHETTA in cui questo indirizzo può essere suddiviso. Definire queste dimensioni per una memoria cache di tipo:

- ad accesso diretto
- 2-set associativa
- completamente associativa

dim. memoria 128K

dim blocco 64

dim. RAM 16M

$$\text{indirizzo RAM} = \log_2 16M \rightarrow \log_2 2^4 \cdot 2^{20} = 2^{24} = 24 \text{ bit}$$

accesso diretto:

$$\text{parole} = \log_2 64 = 2^6 = 6 \text{ bit}$$

$$\text{blocco} = \log_2 \frac{\text{dim. Cache}}{\text{dim. blocco}} = \log_2 \frac{2^{12}}{2^6} = 11 \text{ bit}$$

$$\text{etichetta} : 24 - 11 - 6 = 7 \text{ bit}$$

7 bit	11 bit	6 bit
etichetta	blocco	parola

2-set associativo:

$$\text{parole} = 6 \text{ bit}$$

$$\text{set} = \log_2 \frac{\text{dim. Cache}}{\text{dim. blocco} \cdot \text{set}} = \log_2 \frac{2^{12}}{2^6 \cdot 2^1} = 10 \text{ bit}$$

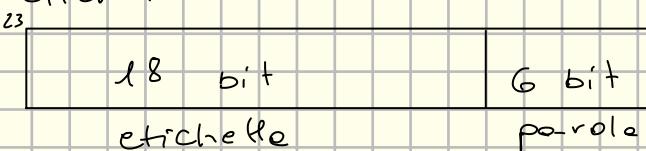
$$\text{etichetta} = 24 - 10 - 6 = 8 \text{ bit}$$

8 bit	10 bit	6 bit
etichetta	set	parola

completamente associativa:

$$\text{parole} = 6 \text{ bit}$$

$$\text{etichette} = 24 - 6 = 17 \text{ bit}$$



Esercizio 4

Si consideri una CPU con una pipeline a 4 stadi (F, D, E, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e che la condizione del salto sia falsa.

istr 1 2 3 4 5 6 7 8 9 10 11

cmpl %eax, %ebx	F D E W
jz START	F D D D E W
subl %ebx, %ecx	F F F D E W
movl %edx, DATA	F D E W

Esercizio 4

Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che il salto condizionato sia di tipo diretto:
JZ (%EAX)+%SI

- 1) PCout, MARin, READ, SELECT_A, ADD, Z_{IN}
- 2) WIFC, Zout, PCin
- 3) MDRout, RDin
- 4) EAXout, MARin, READ
- 5) WIFC, SIout, V_{IN}
- 6) MDRout, SELECT_V, ADD, Z_{IN}
- 7) Zout, PCin, END

Esercizio 4

Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia TRE BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indietro a registro e che il salto sia di tipo relativo:

CALL (%EAX)

- 1) PCout, MARin, READ, SELECT_A, ADD, Z_{IN}
- 2) WIFC, Zout, PCin
- 3) TDRout, RDin
- 4) ESPout, SELECT_A, SUB, Z_{IN}
- 5) Zout, MARin, ESPin
- 6) TDRout, PCin, WRITE
- 7) WIFC
- 8) EAXout, MARin, READ
- 9) WIFC, PCout, V_{IN}
- 10) MDRout, SELECT_V, ADD, Z_{IN}
- 11) Zout, PCin, END

Esercizio 3

- Si assuma che un computer con memoria virtuale, strutturata a pagine di 4Kbyte, sia dotato di 64Mbyte di memoria fisica.

Si consideri un programma con un codice di 7.2Kbyte che accede ciclicamente in sequenza a tutti gli elementi di un array di 1000 record in cui ogni campo è composto da 2 numeri interi. Quale deve essere la dimensione del *working set* perché si abbiano dei *page fault* solamente nella fase di caricamento del programma con esecuzione del primo ciclo di accesso agli elementi dell'array? Quanti *page fault* si avrebbero durante l'esecuzione del programma ipotizzando che il *working set* abbia una pagina meno di quanto definito nel punto precedente e che il ciclo di accesso venga ripetuto 10 volte?

pagine 4 kB → codice 7.2 kB (2 pagine)

dim. Mem. = 64 MB

dim. Array = 1000

$$\text{dati} = 1000 \cdot 4 \cdot 2 = 8000 \text{ b}$$

$$\frac{8000 \text{ byte}}{4 \text{ kB}} = 1,99 \rightarrow \text{occupa 2 pagine}$$

Servono quindi 4 pagine

se ne avessimo 3

4 page fault 1° ciclo

$$2^0 \cdot 3^0 = 2 \cdot 9 = 18$$

22 page fault totali

Esercizio 4

Elencare le micro istruzioni relative alla completa esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola e che (%EBX) rappresenti un metodo di indirizzamento indiretto a registro:

ADD(%EBX), %EAX

1 bus

- 1) PC_{out}, MAR_{IN}, READ, SELECT₊, ADD, Z_{IN}
- 2) W7FC, Z_{OUT}, PC_{IN}
- 3) MD_{OUT}, IR_{IN}
- 4) EBX_{OUT}, MAR_{IN}, READ
- 5) W7FC, EAX_{OUT}, V_{IN}
- 6) MD_{OUT}, SELECT_v, ADD, Z_{IN}
- 7) Z_{OUT}, EAX_{IN}, END

3 bus

- 1) PC_{out B}, R=B, MAR_{IN 2}, READ, INC-PC
- 2) W7FC
- 3) MD_{OUT B}, R=B, IR_{IN 2}
- 4) EBX_{OUT B}, R=B, MAR_{IN 2}, READ
- 5) W7FC
- 6) SELECT[BUS-A], MD_{OUT A}, EAX_{OUT B}, ADD, EAX_{IN 2}, END

Esercizio 3

Si consideri una CPU con una pipeline a 4 stadi (F, D, E, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

istr	1	2	3	4	5	6	7	8	9	10	11	12	13
loop: addl %eax, %ebx	F	D	E	W									
movl ind,%ecx	F	D	E	W									
subl %ebx,%ecx	F	D	D	E	W								
jz loop	F	F	D	D	D	E	W						
movl %ecx,ind	F	F	F	D	E	W							

Esercizio 2 Prima parte

Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia **tre BUS**, che l'istruzione sia composta da una sola parola e che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro (usare solamente le righe necessarie):
JMP (%EAX)

- 1) PC_{OUT} B, R = B, TAD_{IN} R, READ, INC - PC
- 2) WIFC
- 3) MDROUT B, R = B, IR_{IN} R
- 4) EAX_{OUT} B, R = B, MAD_{IN} R, READ
- 5) WIFC
- 6) MDROUT B, R = B, PC_{IN} R, END

Esercizio 2 Seconda parte

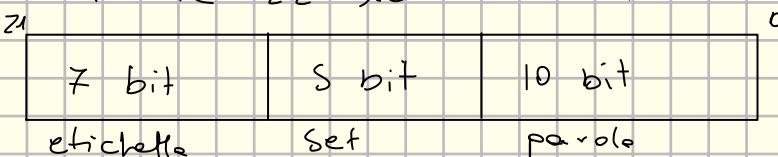
- Si consideri una memoria cache 2-set associativa della dimensione di 64 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 4Mbyte indirizzabile per byte. Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

$$\text{indirizzo RAM} = \log_2 4 \cdot 2^{20} = \log_2 2^2 \cdot 2^{20} = 2^{22} = 22 \text{ bit}$$

$$\text{parole} = 2^{10} \rightarrow \log_2 2^{10} = 10 \text{ bit}$$

$$\text{set} = \log_2 \frac{\text{dim. Cache}}{\text{dim. blocco} \cdot \text{set}} = \frac{2^6 \cdot 2^{10}}{2^{10} \cdot 2^1} = \frac{2^{16}}{2^{11}} = 5 \text{ bit}$$

$$\text{etichetta} = 22 - 10 - 5 = 7 \text{ bit}$$

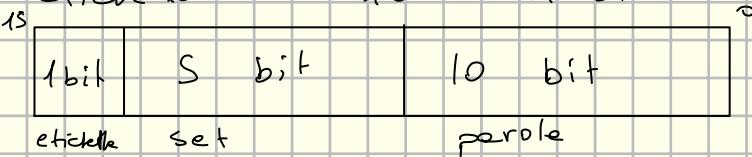


$$\text{indirizzo Cache} = \log_2 2^6 \cdot 2^{10} = \log_2 2^{16} = 16 \text{ bit}$$

$$\text{parole} = 10 \text{ bit}$$

$$\text{set} = 5 \text{ bit}$$

$$\text{etichetta} = 16 - 10 - 5 = 1 \text{ bit}$$



Esercizio 3 Prima parte

Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia relativo al PC:
JNZ (%EAX)

- 1) PC_{OUT}, MAD_{IN}, READ, SELECT+, Z_{IN}, ADD
- 2) WIFC, Z_{OUT}, PC_{IN}
- 3) MDROUT, IR_{IN}
- 4) if(ZERO) END, EAX_{OUT}, MAD_{IN}, READ
- 5) WIFC, PC_{OUT}, V_{IN}
- 6) TDR_{OUT}, SELECT+, ADD, Z_{IN}
- 7) Z_{OUT}, PC_{IN}, END

Esercizio 4

Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%Exx) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia assoluto:
JNZ (%EAX + %EBX)

- 1) PC_{OUT}, M_{AD}_{IN}, READ, SELECT_A, ADD, Z_{IN}
- 2) W_{IFC}, Z_{OUT}, PC_{IN}
- 3) M_{DATA}_{OUT}, I_R_{IN}
- 4) EAX_{OUT}, U_{IN}
- 5) EBX_{OUT}, SELECT_B, ADD, Z_{IN}
- 6) Z_{OUT}, M_{AR}_{IN}, READ
- 7) W_{IFC}
- 8) TDR_{OUT}, PC_{IN}, END

Esercizio 4

Elencare e commentare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%Exx) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia assoluto:

CALL (%EAX + %EBX)

- 1) PC_{OUT}, M_{AD}_{IN}, READ, SELECT_A, ADD, Z_{IN}
- 2) W_{IFC}, PC_{IN}, Z_{OUT}
- 3) M_{DATA}_{OUT}, I_R_{IN}
- 4) ESP_{OUT}, SELECT_A, SUB, Z_{IN}
- 5) Z_{OUT}, ESP_{IN}, M_{AR}_{IN}
- 6) M_{DATA}_{OUT}, PC_{IN}, WRITE
- 7) W_{IFC}, EAX_{OUT}, U_{IN}
- 8) EBX_{OUT}, SELECT_B, ADD, Z_{IN}
- 9) Z_{OUT}, M_{AR}_{IN}, READ
- 10) W_{IFC}
- 11) M_{DATA}_{OUT}, PC_{IN}, END

Esercizio 2

Si consideri una memoria cache 4-set associativa della dimensione di 32 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 4Mbyte indirizzabile per byte.

- Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

$$\text{indirizzo RAM} = \log_2 2^2 \cdot 2^{20} = \log_2 2^{22} = 22 \text{ bit}$$

parole = $\log_2 2^{10} = 10 \text{ bit}$

$$\text{set} = \log_2 \frac{\text{dim. Cache}}{\text{dim blocco} \cdot \text{set}} = \log_2 \frac{2^5 \cdot 2^{10}}{2^{10} \cdot 2^2} = 3 \text{ bit}$$

$$\text{etichette} = 22 - 3 - 10 = 9 \text{ bit}$$

9 bit	3 bit	10 bit
etichette	set	parole

$$\text{indirizzo Cache} = \log_2 32K = \log_2 2^5 \cdot 2^{10} = \log_2 2^{15} = 15 \text{ bit}$$

parole e set sono uguali

$$\text{etichette} = 15 - 10 - 3 = 2 \text{ bit}$$



- 3) Si consideri una memoria cache 4-set associativa della dimensione di 16 Kbyte con 512 byte per blocco. La cache è collegata ad una memoria di 2Mbyte indirizzabile per byte.

- Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

$$\text{indirizzo RAM} = \log_2 2^4 \cdot 2^{10} = \log_2 2^{14} = 14 \text{ bit}$$

$$\text{parola} = \log_2 512 = 9 \text{ bit}$$

$$\text{cache} = 2^4 - 4 - 3 = 3 \text{ bit}$$

3 bit	3 bit	9 bit
etichette	set	parola

$$\frac{2^4 \cdot 2^{10}}{2^9 \cdot 2^2} = 3 \text{ bit}$$

$$\text{indirizzo Cache} = \log_2 2^4 \cdot 2^{10} = \log_2 2^{14} = 14 \text{ bit}$$

$$\text{etichette} = 14 - 9 - 3 = 2 \text{ bit}$$

2 bit	3 bit	9 bit
etichette	set	parola

trovare l'etichetta

- Si assuma che la cache appena descritta sia utilizzata per memorizzare i dati di un programma. Si assume che sia inizialmente vuota e che utilizzi un algoritmo di sostituzione dei blocchi di tipo LRU (sostituzione dell'elemento meno utilizzato di recente).

Il programma di benchmark accede in sequenza a tutti gli elementi di un array di 2000 numeri interi memorizzato a partire dall'indirizzo 0. Si ricorda che ogni intero ha una dimensione di quattro byte. Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 5 volte. Si assume che il tempo di accesso ad un byte della cache sia 1T ed in memoria sia 10T. Calcolare il tempo di accesso ai dati in presenza della cache.

$$N^{\circ} \text{ blocchi tot} = \frac{2000 \cdot 2}{512} = \frac{4000}{512} \approx 7,8 \rightarrow \text{servono 8 blocchi}$$

$$N^{\circ} \text{ blocchi Cache} = 32 \text{ bit}$$

$$T_{sc} = 4000 \cdot 5 \cdot 10 = 200000$$

$$1^{\circ} \text{ Ciclo} = 8 \text{ miss}$$

$$2^{\circ} - 5^{\circ} \text{ Ciclo} = 0 \text{ miss}$$

$$\text{blocchi tot letti da RAM} = 8$$

$$\text{parole lette da RAM} = 8 \cdot 512 = 4096$$

$$\text{parole tot lette da Cache} = 64 \cdot 8 \cdot 5 - 4096 = 512$$

$$T_{cc} = 4096 \cdot 10 + 512 = 41472$$

$$\text{Speedup} = \frac{200000}{41472} = 4,8$$

- Si consideri un programma che accede in sequenza a tutti gli elementi di un array di 10000 parole (ogni elemento ha le dimensioni di una parola). Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 10 volte.

Si assume che le dimensioni di una pagina siano di 1K parole. Il programma occupa due pagine ed il suo *working set* è composto da 10 pagine. Quanti *page fault* avvengono durante l'esecuzione del programma considerando che nessuna pagina del programma è residente in memoria al momento della sua esecuzione?

dim. array = 10000

ciclo = 10

pagina = 1K codice 2 pagine WS = 10 pagine

pagina = 1024

$$\frac{10000}{1024} = 9,7 \rightarrow \text{Saranno 10 pagine per contenarlo}$$

1° ciclo 12 page fault

2°-10° ciclo $10 \cdot 9 = 90$ page fault

102 page fault totali

- Si consideri un numero in virgola mobile in singola precisione.

- Specificare il significato dei 32 bit che lo compongono.

il primo bit indica il segno 1(-), 0(+) → bit 31 (+ significativo)
gli 8 bit a destra sono l'esponente → esponente del numero + bias, che in singola precisione è 127

i rimanenti 23 sono la mantissa → numero modificato in virgola fissa con la virgola spostata per avere 1, ...

- Qual è il numero più grande in virgola mobile che può essere rappresentato?

0 111111 00000000000000000000000000000000 → + ∞

Se non ci fossero solo 0 avremmo errore

NaN (Not a Number)

- A quale numero decimale corrisponde il numero binario in virgola fissa 01010011.01110

$$01010011_2 \rightarrow 1+2+16+64=83_{10}$$
$$01110 \rightarrow \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = \frac{4+2+1}{16} = \frac{7}{16}_{10} \rightarrow 01010011.01110_2$$
$$(83 + \frac{7}{16})_{10}$$

- Quali sono i vantaggi della rappresentazione in virgola mobile rispetto a quella in virgola fissa?

Possedere dello virgola fissa allo mobile ci permette di codificare i numeri senza sapere quanti bit dedicare alla parte intera e alle porle frazionarie.

- 3) Si consideri una memoria cache 4-set associativa della dimensione di 32 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 1Mbyte indirizzabile per byte. Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

$$\begin{aligned} \text{indirizzo RAM} &= \log_2 1M = \log_2 2^{20} = 20 \text{ bit} \quad \text{parola} = \log_2 1024 = \log_2 2^{10} = 10 \text{ bit} \\ \text{Set} &= \log_2 \frac{\text{dim Cache}}{\text{dim blocco} \cdot \text{set}} = \frac{2^{10}}{2^9 \cdot 2^2} = 3 \text{ bit} \quad \text{etichetta} = 20 - 10 - 3 = 7 \text{ bit} \end{aligned}$$

2 bit	3 bit	10 bit
etichette	set	parole

$$\begin{aligned} \text{indirizzo Cache} &= \log_2 32K = \log_2 2^5 \cdot 2^{10} = 2^{15} = 15 \text{ bit}, \text{ campi parola e set sono uguali} \\ \text{etichette} &= 15 - 10 - 3 = 2 \text{ bit} \end{aligned}$$

2 bit	3 bit	10 bit
etichette	set	parole

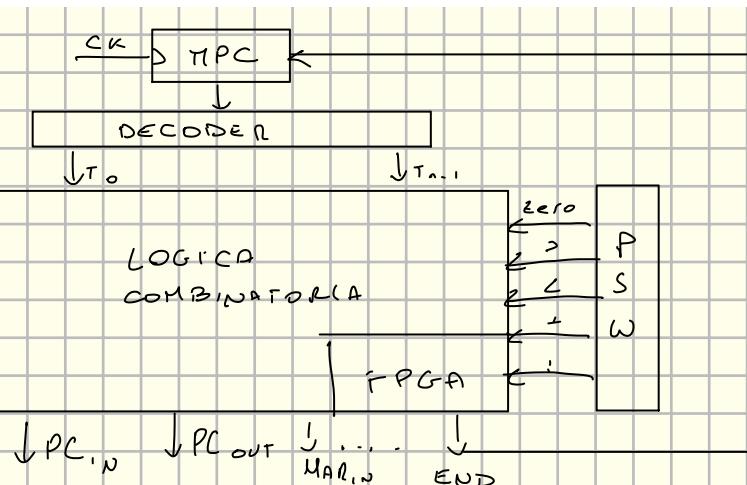
- Quali sono i vantaggi e gli svantaggi delle memorie completamente associative rispetto alle memorie non associative.

Le memorie completamente associative, a differenza delle non associative hanno il vantaggio di poter inserire il dato in un indirizzo non prevedibile, dando così libertà nell'utilizzo delle memorie e cercando conflitti solo quando la memoria è piena. Gli svantaggi sono i costi elevati e l'obbligo di un algoritmo per trovare i blocchi.

- Quali sono le motivazioni che fanno preferire la realizzazione di unità di controllo cablate rispetto a quelle microprogrammate.

Le unità cablate, rispetto a quelle microprogrammate sono le prestazioni. Le unità cablate hanno una specifica ISA che può essere modificata tramite FPGA, dove si trovano nel blocco di logica combinatoria, che può essere modificato. Le unità microprogrammate avevano una mem. con una serie di parole. Si adattano di più a situazioni diverse ma come detto hanno prestazioni inferiori.

- Definire lo schema di un controllore cablato indicando i segnali utilizzati e la funzione dei blocchi presenti.



Il microprogram counter ha il compito di generare dei segnali che sono i possi di controllo. Il suo decoder fa passare il segnale i possi di controllo dove T_{n-1} è il massimo n° di possi di controllo per svolgere le operazioni.

Il de l'istruzione corrente ed il decoder fa passare quelle da eseguire. END è il segnale di RESET del MPC. PSW de bit corrispondi specifici alla logica combinatoria. La logica combinatoria opera e crea i vari segnali: PCIN, PCOUT...

- 2) Descrivere su quali principi si basa il metodo di Quine-McKluskey per la minimizzazione esatta di funzioni combinatorie ad una uscita completamente specificate.

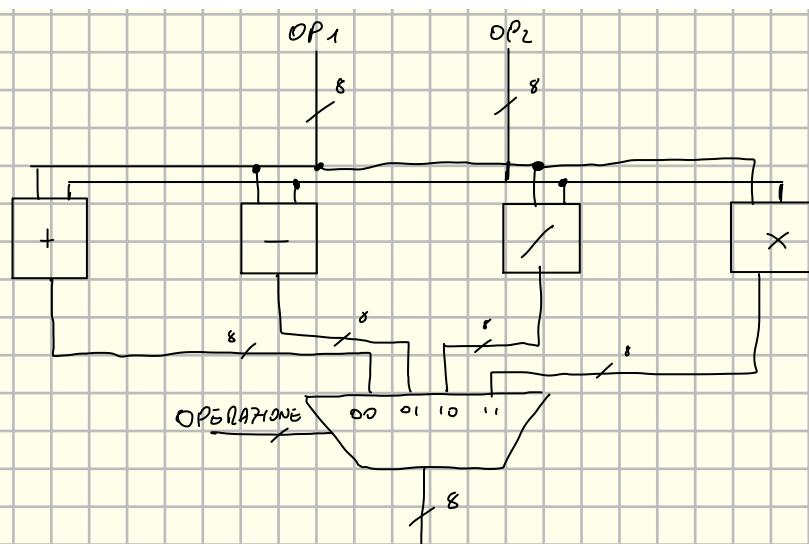
Il metodo di Q&MK si basa su 2 fattori, trovare implicanti primi e la copertura massima. 1) TROVARE GLI IMPLICANTI: Si rappresenta l'insieme in ordine a distanza Hamming 1, si raggruppano quelli che hanno il solo bit di diff. sostituendolo con un DC. Si ripete lo stesso passaggio cercando quelli che hanno il solo bit di diff e DC nella stessa pos. Tutti i mintermini che non sono stati uniti durante i passaggi saranno Implicanti primi.

Ora si deve trovare la copertura ottima tramite dominanza (per righe o colonne dipende da come si fa la tabella). Quelli che rimarranno saranno I. p. essenziali.

- Come si può estendere il metodo al caso di funzioni parzialmente specificate?

Si fa come appena descritto però si utilizza il DC-SET come mintermini nella ricerca degli implicanti e come Nextermimi nella copertura ottima

- Progettare, utilizzando componenti di libreria, una ALU che può eseguire le quattro operazioni aritmetiche (somma, sottrazione, moltiplicazione, divisione) su due operandi a 8 bit.



- 4) Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola e che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro (usare solamente le righe necessarie): *assomando salto ce l'atrio*

CALL (%EAX)

- 1) PCout, MDRin, READ, SELECT4, ADD, Zin
- 2) WIFC, PCin, Zout
- 3) IDOUT, IRin
- 4) ESPout, SELECT4, SUB, Zin
- 5) Zout, MDRin, ESPin
- 6) PCout, MDRin, WRITE
- 7) WIFC
- 8) EAXout, MDRin, READ
- 9) WIFC, PCout, Vin
- 10) IDOUT, SELECT4, ADD, Zin
- 11) Zout, PCin, END

Se la CPU avesse 3 bus in questo caso specifico non avrebbe vantaggi, rimarrebbero comunque 11 fasi. Generalmente però poter avere più bus ci permette di avere + dati in out e quindi velocizzare le operazioni.

AB_{16} a quelli in interi può corrispondere?

$A B_{16}$

$1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1$
2

Modulo : $1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1$ $\rightarrow 1 + 2 + 8 + 32 - 128 = 171_{10}$
 $128 \quad 32 \quad 8 \quad 2 \quad 1$

Modulo e Segno $\rightarrow 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \rightarrow 1 + 2 + 8 + 32 = -43_{10}$

Complemento a due $\rightarrow 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ +$
 $1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1$
 $\hline 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \rightarrow 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 = -87_{10}$

Vantaggi e svantaggi del C+2:

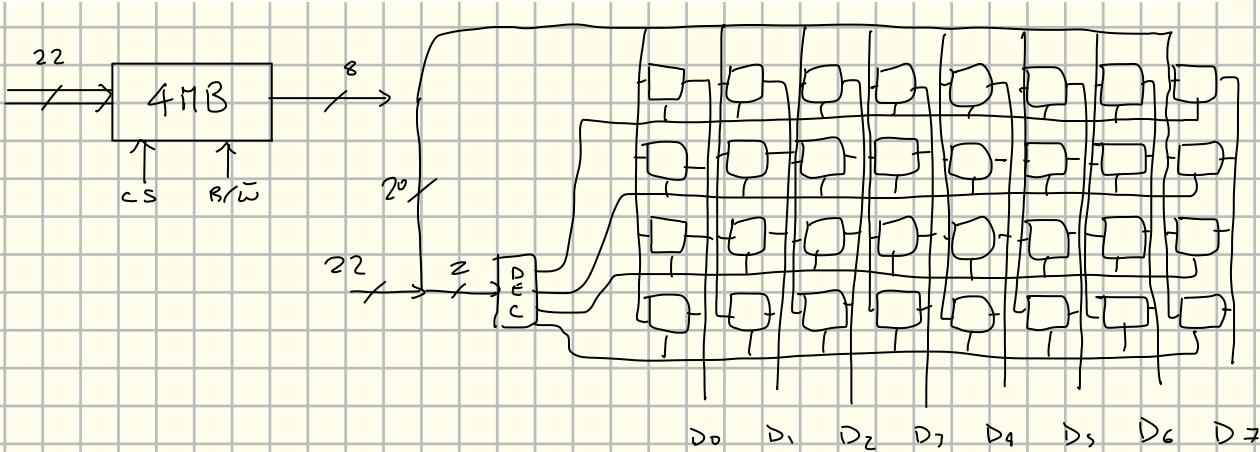
I vantaggi sono che non c'è il problema del doppio zero (± 0) e che si semplificano le op. aritmetiche, è stato ideato infatti partendo dal presupposto che $-1+1=0$.

$$\begin{array}{r} 00001 \\ ??\ ?\ ?\ ? \\ \hline 00000 \end{array} \quad \begin{array}{r} 00001 \\ 11111 \\ \hline 00000 \end{array}$$

Gli svantaggi sono che avendo un range $-2 \leq n \leq 2^{n-1}$ ciò rende impossibile rappresentare i dati corrispondenti positivi del numero più piccolo ed esempio su 8 bit il -128 non ha corrispondenza positiva. C'è anche il problema della gestione dell'overflow quando con un'operazione si esce dai limiti del range sopra sentito \rightarrow ciò richiederebbe la possibilità di dover avere logiche aggiuntive.

3) Si consideri una memoria statica di 4Mbyte indirizzabili un byte alla volta.

- Definire le dimensioni dei segnali di dato e indirizzamento e disegnare lo schema del chip di memoria avendo a disposizione moduli da 1Mbit.



- 3) Si consideri un sistema a memoria virtuale con spazio logico di 4G parole, una memoria fisica di 32M parole e dimensione delle pagine di 16K parole.

- Determinare il numero di bit che definiscono:

Lunghezza indirizzo logico:	32 bit	$\leftarrow \log_2 2^{30} = 32 \text{ bit}$
Di cui per Num. Pag. Logica	18 bit	$\leftarrow \log_2 \frac{2^{32}}{2^{14}} = 18$
per lo spiazzamento	14 bit	$\leftarrow \log_2 2^4 = 14$
Lunghezza indirizzo fisico	25 bit	$\leftarrow \log_2 2^5 \cdot 2^{20} = 25 \text{ bit}$
di cui per Num. Pag. Fisica	11 bit	$\leftarrow \log_2 \frac{2^{25}}{2^{14}} = 11$
per lo spiazzamento	14 bit	

- Nella seguente tabella sono riportati alcuni valori del parametro R (numero di pagine residenti per processo); sapendo che il Sistema Operativo occupa permanentemente 448 pagine e che sono stati creati 22 processi, indicare per ogni valore di R il numero di processi in stato di "fuori memoria" ossia che non possono avere tutte le pagine in memoria.

R	40	80	160	320	800
Numero di processi fuori memoria	0	2	12	17	20

- Specificare come è stato calcolato il valore per R = 160:

$$\textcircled{a}) 2^{11} = 2048 - 448 = 1600$$

$$\textcircled{b}) \frac{1600}{160} = 10 \rightarrow 22 - 10 = 12$$

↓
processi
ammisibili
in memoria

- Si supponga di avere un sistema basato su memoria virtuale configurato con R=10 e di sapere che il valore massimo di k (numero di accessi) per il quale il Working Set W(k)=10 è 1000, cioè $W(k) > 10$ per $k > 1000$. In base a queste ipotesi è possibile valutare i valori massimo e minimo del numero di page fault, delle percentuali di page fault e delle percentuali di successo (Hit Rate). Definire tali valori con una breve spiegazione.

- $W(k) = 10$ finché $k \leq 1000 \rightarrow$ cioè tiene in mem fino a 10 pagine consecutivi

- $k = 1000$ mille processi consecutivi

n° page fault = 10

% page fault = $10 / 1000 = 1\%$ } page fault minimi
Hit rate = 99%

n° page fault: 1000

$$\% \text{ di page fault} = 1000 = 100\% \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{page fault massimi}$$

Hit rate = 0%

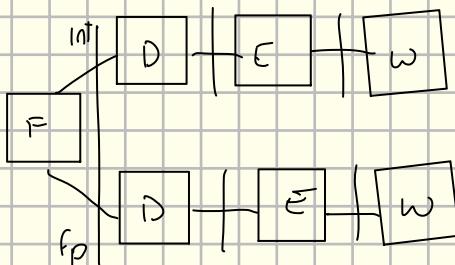
- 4) Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia relativo al PC (usare solamente le righe necessarie):

CALL (%EAX)

- 1) $Z_{IN}, MAR_{IN}, READ, SELECT_4, ADD, Z_{IN}$
- 2) $WIFC, Z_{OUT}, PC_{IN}$
- 3) MDR_{OUT}, IR_{IN}
- 4) $ESP_{OUT}, SELECT_4, SUB, Z_{IN}$
- 5) $Z_{OUT}, ESP_{IN}, MAR_{IN}$
- 6) $PC_{OUT}, MDR_{IN}, WRITE$
- 7) $WIFC$
- 8) $EAX_{OUT}, MAR_{IN}, READ$
- 9) $WIFC, PC_{OUT}, V_{IN}$
- 10) $MDR_{OUT}, SELECT_4, ADD, Z_{IN}$
- 11) Z_{OUT}, PC_{IN}, END

- Definizione di microprocessore superscalare e come sia possibile realizzarlo

Le architetture superscalari sono quelle che hanno $CPI_m < 1$. Si realizzano utilizzando due doppie pipeline con un solo fetch. Lo fetch divide i flussi che lavorano sugli interi e sui n. in virgola mobile.



- Aumenta la frequenza degli interi rispetto a quelli in virgola fissa.
- Se non avessero bolle o fine pipeline avremo un $CPI_m = 0,5$ istr in un ciclo di CK.

C'è però un limite dato dalle dipendenze e queste non possono essere modificate o "cause" di come sono scritti gli algoritmi

- 1) Elencare le micro istruzioni relative alla completa esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola e che (%EBX) rappresenti un metodo di indirizzamento indiretto a registro (usare solamente le righe necessarie):

ADDL (%EBX), %EAX

- 1) $PC_{OUT}, MAR_{IN}, READ, SELECT_4, ADD, Z_{IN}$
- 2) $WIFC, Z_{OUT}, PC_{IN}$
- 3) MDR_{OUT}, IR_{IN}
- 4) $EBX_{OUT}, MAR_{IN}, READ$
- 5) $WIFC, EAX_{OUT}, V_{IN}$
- 6) $MDR_{OUT}, SELECT_4, ADD, Z_{IN}$
- 7) Z_{OUT}, EAX_{IN}, END

- L'esecuzione di un programma compilato per una CPU antiquata da parte di una CPU più moderna, dotata solamente di un numero maggiore di registri generali, può migliorare i tempi di esecuzione?

Il miglioramento può esserci perché avere più registri personali e modo di avere meno accessi alla memoria velocizzando così i processi. Naturalmente questo miglioramento si presenterà se il codice sfrutterà questo vantaggio.

- 2) Si consideri una CPU dotata di memoria cache 4-associativa di 8K parole con 64 parole per blocco. Questa CPU è collegata ad una memoria RAM da 4M parole.

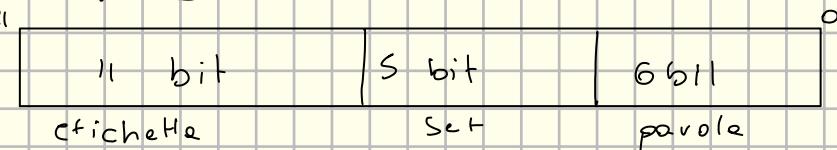
- Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO ed ETICHETTA in cui questo indirizzo può essere suddiviso. Motivare la risposta con un opportuno schema.

$$\text{indirizzo RAM} = \log_2 \text{dim. RAM} = \log_2 2^? \cdot 2^{20} = \log_2 2^{22} = 22 \text{ bit}$$

$$\text{parole} = \log_2 64 = \log_2 2^6 = 6 \text{ bit}$$

$$\text{set} = \log_2 \frac{\text{dim. Cache}}{\text{parole} \cdot \text{set}} = \log_2 \frac{2^3 \cdot 2^{10}}{2^6 \cdot 2^2} = \log_2 \frac{2^{13}}{2^8} = 5 \text{ bit}$$

$$\text{etichetta} = 22 - 5 - 6 = 11 \text{ bit}$$



- Si assuma che la cache appena descritta sia utilizzata per i dati, che sia inizialmente vuota e che utilizzi un algoritmo di sostituzione dei blocchi di tipo LRU (sostituzione dell'elemento meno utilizzato di recente). La CPU esegue un programma che accede in sequenza a tutti gli elementi di un array di 8320 parole (ogni elemento ha le dimensioni di una parola) che è memorizzato a partire dall'indirizzo 0. Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 5 volte.

Si assume che il tempo di accesso alla cache sia di 1T e che il tempo di accesso alla memoria sia di 10T (entrambi i tempi si riferiscono alla lettura di una parola). Calcolare il rapporto (fattore di miglioramento) tra il sistema in presenza di cache e in assenza di cache per l'esecuzione di questo programma.

dim cache : 8K

parole : 64

dim. RAM = 4M

Set = 4

dim. Array = 8320

ciclo S (LRU)

Tc = 1T

Tmem = 10T

$$\begin{aligned} n^{\circ} \text{ Tot blocchi} &= \frac{8320}{64} = 130 \\ n^{\circ} \text{ blocchi Cache} &= \frac{8192}{64} = 128 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} 2 \text{ blocchi mancanti}$$

$$T_{sc} = 8320 \cdot 5 \cdot 10 = 416000$$

$$1^{\circ} \text{ Ciclo} = 130 \text{ Miss} \rightarrow \text{bisogna caricare tutto}$$

$$2^{\circ} - 5^{\circ} = [(2 \cdot 4) + 2] \cdot 4 = 40 \text{ Miss}$$

$$\text{Totale blocchi letti da RAM} = 170 \text{ Miss}$$

$$\text{Totale parole lette da RAM} = 170 \cdot 64 = 10880$$

$$\text{Totale parole lette da cache} = (130 \cdot 64 \cdot 5) - 10880 = 30720$$

$$T_{cc} = 10880 \cdot 10 + 30720 = 139520$$

$$\text{Speedup} = \frac{416000}{139520} = 2,98$$

4) Rispondere alle seguenti domande riportando la motivazione della risposta.

- Una CPU con una pipeline a 2 stadi viene sostituita con una CPU con una pipeline a 4 stadi. Se il tempo totale di esecuzione di una singola istruzione è rimasto invariato, qual è il minimo ed il massimo incremento delle prestazioni che si può attendere?

Il massimo incremento è il doppio delle prestazioni, nel caso peggiore nessun miglioramento (caso di multiple dipendenze)

- Si consideri una CPU con una pipeline a 4 stadi (F, D, O, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

clock/istruzione	1	2	3	4	5	6	7	8	9	10	11
addl %eax, %ebx	F	D	E	W							
movl ind, %ecx		F	D	E	W						
subl %ebx, %ecx			F	D	D	D	E	W			
jz loop				F	F	F	D	D	D	E	W

- Descrivere il meccanismo e l'utilità della *predizione dei salti*.

Il meccanismo della predizione di salto (branch prediction): si predice in anticipo se il salto ovvero:

- Se avviene, allora l'istruzione prossima è quella di destinazione del salto
- Se non avviene l'istruzione prossima è quelle successive in ordine.

Se non avviene si scarta l'istruzione appena iniziata.

Esistono 2 tipologie:

- statico: lo decide il compilatore

- predizione fissa: salta sempre o non salta mai
- predizione basata sullo spostamento delle Branch: se positivo salta in avanti se no indietro. Generalmente non ha ragione nei salti in avanti.

- dinamico: controlla lo storico dei salti di un'istruzione, si conserva lo stato di uno FSM che definisce l'evoluzione

Es 10101100

$$- \text{MODULO} : 10101100_2 = 172_{10}$$

$$- \text{MODULO E SEGNO} = 10101100_2 = -44$$

$$- \text{COMPIMENTO + DUE} = 10101100 + \\ \begin{array}{r} 1111111 \\ \hline 10101011 \end{array}$$

$$10101011 \rightarrow 01010100_2 = -84$$

$$\text{MODULO: } 10101100_2 = 172_{10}$$

$$01010101011_2 \rightarrow 363_{10}$$

$$363 + 172 = 535_{10}$$

$$\begin{array}{r} 111111 \\ 01010101011 + \\ 0010101100 \\ \hline 1000010111 \end{array}$$

$$1000010111_2 = 535_{10}$$

MODULO E SEGNO: $363 + (-44) = 319_{10}$

$$\begin{array}{r}
 & \overset{0}{\cancel{1}} \overset{0}{\cancel{0}} \\
 1 & 0 & \cancel{1} & \cancel{1} & \overset{10}{\cancel{0}} & 1 & 1 & - \\
 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
 \hline
 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 = 319_{10}
 \end{array}$$

COMPLEMENTO A DUE: $-84_{10} + 363_{10} = 279_{10}$

$$\begin{array}{r}
 & \overset{1}{\cancel{1}} \overset{1}{\cancel{1}} \overset{1}{\cancel{1}} \\
 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & + \\
 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
 \hline
 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & = +279_{10}
 \end{array}$$

2) $f(a, b, c, d)$

ON-SET = {m₂, m₅, m₈, m₉, m₁₀, m₁₅}
DC-SET = {m₁, m₃, m₆, m₇, m₁₁, m₁₃}

a b c d	0	m ₁	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅
0 0 0 0	0	m ₀	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅
0 0 0 1	-	m ₁	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	-	I ₅
0 0 1 0	1	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	I ₆
0 0 1 1	-	m ₃	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	I ₇
0 1 0 0	0	m ₆	m ₅	m ₄	m ₃	m ₂	m ₁	m ₀	m ₁₅	m ₁₄	m ₁₃	m ₁₂	m ₁₁	m ₁₀	m ₉	I ₈
0 1 0 1	1	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	I ₉
0 1 1 0	-	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	I ₁₀
0 1 1 1	-	m ₇	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	-	-	-	I ₁₁
1 0 0 0	1	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	-	-	I ₁₂
1 0 0 1	1	m ₉	m ₇	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	-	-	-	I ₁₃
1 0 1 0	1	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	-	-	-	-	I ₁₄
1 0 1 1	-	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	-	-	-	-	-	I ₁₅
1 1 0 0	0	m ₁₂	m ₁₃	m ₁₄	m ₁₅	-	-	-	-	-	-	-	-	-	-	I ₁₆
1 1 0 1	-	m ₁₃	m ₁₅	-	-	-	-	-	-	-	-	-	-	-	-	I ₁₇
1 1 1 0	0	m ₁₄	m ₁₅	-	-	-	-	-	-	-	-	-	-	-	-	I ₁₈
1 1 1 1	1	m ₁₅	-	-	-	-	-	-	-	-	-	-	-	-	-	I ₁₉

~~I₀~~ I₁ I₂ ~~I₃~~ I₄ ~~I₅~~ I₆ ~~I₇~~ I₈ ~~I₉~~ I₁₀ ~~I₁₁~~

m_L

1 1

m_S

1 1 1

m₈ , ,

1 1 1

m₅

1 1 1

m₁₀ 1 1

1

m₁₅

1 1 1

I₁ I₆ I₈ I₁₀

m2

m5

m8

m9

m10

m15

