CORSO DI
ARCHITETTURA DEGLI ELABORATORI
Corso di Laurea in Informatica

DISPENSA

Ing. Alfredo Garro
Prof. Pasquale Rullo

ESERCIZIO: progettazione di una CPU didattica

Si vuole progettare una C.P.U capace di eseguire programmi scritti usando le istruzioni indicate in tabella. La macchina deve disporre di un registro speciale detto Accumulatore (**Acc**) in cui si trova sempre uno degli operandi richiesti nelle istruzioni. La macchina deve gestire una memoria (**M**) da 4K byte.

ISTRUZIONI del LINGUAGGIO MACCHINA

Istruzione	Semantica
STORE x	Acc→M[x]
LOAD x	M[x]→Acc
ADD x	Acc+M[x]→Acc
SUB x	Acc-M[x]→Acc
JZ x	Se Acc==0 vai all'Ist che si trova all'Ind. x
JUMP x	Vai all'Ist che si trova all'Ind. x

Si noti che tutte le istruzioni hanno un unico operando che è un indirizzo di memoria.

Poiché il numero di istruzioni è pari a 6, sono sufficienti 3 bit per rappresentarle; in particolare, possiamo utilizzare la seguente codifica:

LOAD: 001
STORE: 010
ADD: 011
SUB: 100
JZ: 101
JUMP: 110

Considerato inoltre che la memoria ha una capacità di 4 Kbyte, cioè 4096 byte, ogni indirizzo è lungo 12 bit (2^{12}=4096).

Ne consegue che ogni **istruzione è lunga 15 bit** (3 per codice operativo e 12 per l'indirizzo di memoria).

Quando un programma è in esecuzione, le istruzioni che lo compongono e i dati su cui opera sono memorizzati nella memoria M.
Considerato che ogni istruzione è lunga 15 bit, sono necessari 2 byte per la sua memorizzazione (sono utilizzati i 15 bit più significativi).
Per quanto riguarda i dati, assumiamo che la macchina operi solo su numeri naturali binari ognuno memorizzato in una sequenza di 2 byte ($2^{16} - 1$ è quindi il più grande numero rappresentabile).
Nel seguito utilizziamo il termine CELLA per indicare una sequenza di due byte.

PROGRAMMI IN LINGUAGGIO MACCHINA

Si consideri il seguente frammento di programma scritto in C:

```
…
if a != 0  a = a+b;
else a = a-b;
…..
```

La traduzione di tale frammento nel nostro linguaggio macchina dà luogo al seguente programma:

…
1. LOAD a
2. JZ 5
3. ADD b
4. JUMP 6
5. SUB b
6. STORE a
7. …

```
Const i=1;
Int a = 5;
int b=50;
While a != 0 {
        b=b-a;
        a=a-i;
}
```

1. LOAD a
2. JZ 10
3. LOAD b
4. SUB a
5. STORE b
6. LOAD a
7. SUB i
8. STORE a
9. JUMP 2
10. STOP

```
Const n=10
Const a=1
Int x = 0;
For int i=a to n-1 do
    X= x+a
```

```
            Load a
            Store i
            Load i
Ciclo:      Sub n
```

```
        JZ fine
        Load x
        Add a
        Store x
        Load i
        Add a
        Jump  ciclo
Fine:   halt
```

ESECUZIONE DI PROGRAMMI

Un programma in esecuzione è memorizzato nella memoria centrale M. La CPU opera come segue:

1. carica nel registro IR l'istruzione il cui indirizzo è nel PC;
2. incrementa il PC
3. esegue l'istruzione corrente memorizzata nel registro IR;
4. torna la passo 1

Il suddetto ciclo viene chiamato "ciclo della CPU"; il passo 1 di caricamento della prossima istruzione da eseguire nell'IR è detto "**fetching**"

ARCHITETTURA DELLA PARTE OPERATIVA

Per implementare le istruzioni indicate in tabella la parte operativa della C.P.U dovrà essere composta dai seguenti componenti:

- Registro Acc (Accumulator)
- Registro IR (Instruction Register)
- Registro PC (Program Counter)
- Registro MAR (Memory Address Register)
- Registro MDR (Memory Data Registrer)
- ALU

Il registri MAR e MDR interfacciano la CPU con la memoria M. La dimensione dei registri sopra indicati è la seguente:

- registro MAR: Poiché dobbiamo indirizzare una memoria con 4K byte, cioè 4096 celle, abbiamo bisogno di 12 bit (2^{12}=4096); quindi il registro MAR sarà a 12 bit.
- registro IR: Poiché le istruzioni sono memorizzate in una cella, il registri IR è di 2 byte (viene trascurato solo il bit meno significativo)

- registro PC: siccome contiene **l'indirizzo di memoria** in cui si trova la prossima istruzione che deve essere eseguita, il PC è un registro di 12 bit; il PC è un registro contatore

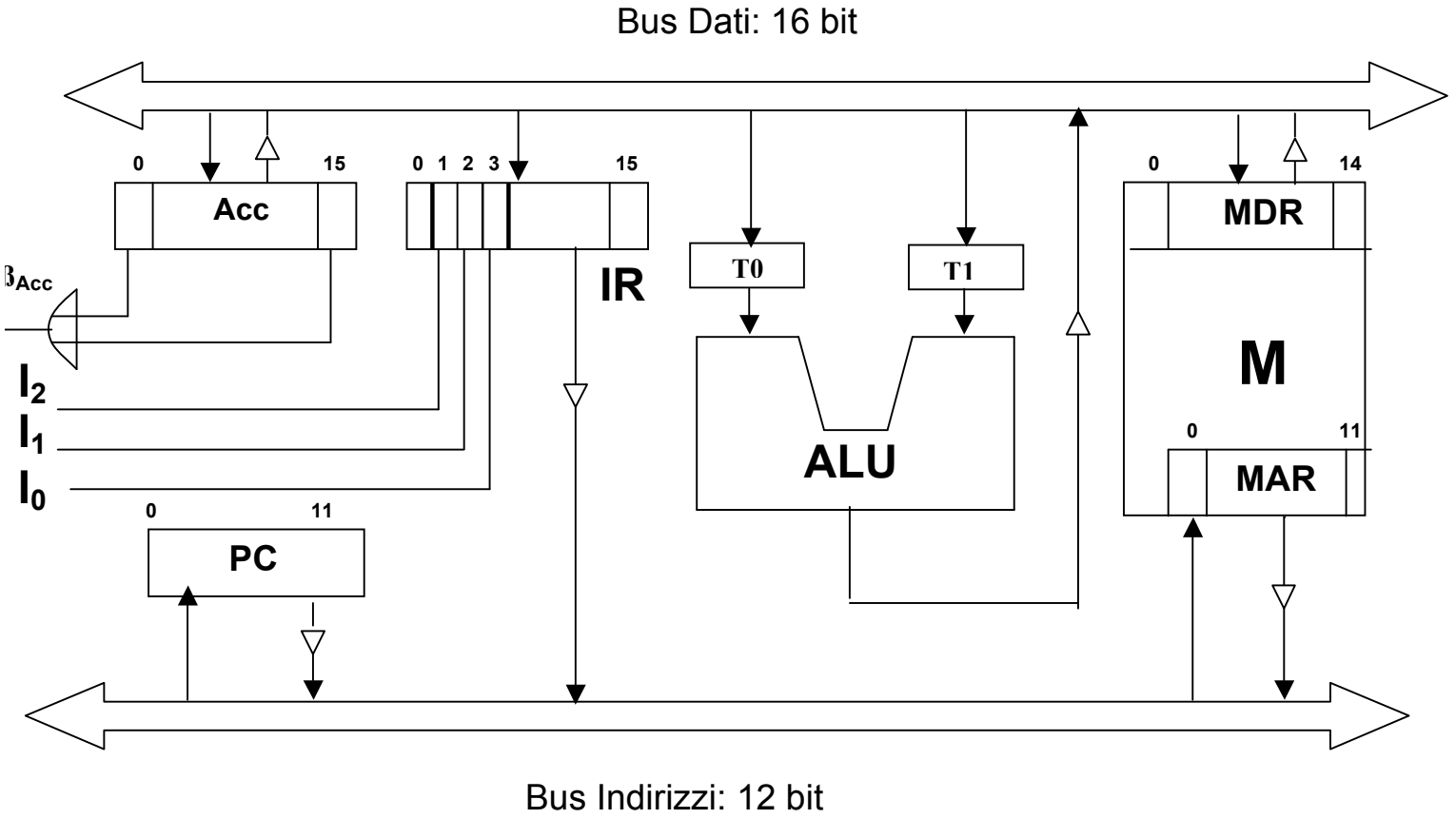
- registro MDR: Poiché le istruzioni e i dati sono memorizzati in una cella di memoria, il registro MDR è anch'esso di 2 byte

- registro Acc: Deve essere in grado di contenere il valore di una cella di memoria, quindi 2 byte.

Da quanto detto si ricava, inoltre, che il bus indirizzi dovrà essere a 12 bit ed il bus dati dovrà essere a 16 bit.

Per verificare che il contenuto del registro Acc sia nullo, come richiesto dall'istruzione JZ x, è necessaria una porta logica che metta in OR tutti i bit del registro; se l'uscita di tale porta sarà 0 allora il valore contenuto in Acc è nullo altrimenti almeno un bit di Acc è pari ad 1 e, quindi, il valore contenuto in Acc è diverso da zero.

Segue lo schema della parte operativa:



SEGNALI DI CONTROLLO DELLA PARTE OPERATIVA

La parte operativa schematizzata in figura è controllata dai seguenti **segnali di controllo**:

- A_{acc} = segnale di abilitazione alla scrittura (caricamento) del registro Acc
- S_{acc} = segnale di abilitazione alla lettura del registro Acc
- A_{T0} = segnale di abilitazione al caricamento del registro T_0
- A_{T1} = segnale di abilitazione al caricamento del registro T_1
- A_{IR} = segnale di abilitazione alla scrittura (caricamento) del registro IR
- S_{IR} = segnale di abilitazione alla lettura del registro IR
- Z_{IR} = segnale di azzeramento del registro IR
- A_{MAR} = segnale di abilitazione alla scrittura (caricamento) del registro MAR
- S_{MAR} = segnale di abilitazione alla lettura del registro MAR
- A_{MDR} = segnale di abilitazione alla scrittura (caricamento) del registro MDR
- S_{MDR} = segnale di abilitazione alla lettura del registro MDR
- A_{PC} = segnale di abilitazione alla scrittura (caricamento) del registro PC
- S_{PC} = segnale di abilitazione alla lettura del registro PC
- K_{PC} = segnale per il controllo dell'incremento del PC (ricorda che PC è un registro contatore)

- AL_0 e AL_1 per l'ALU: se AL_0=0 ed AL_1=0 si ordina una somma, se AL_0=0 ed AL_1=1 una differenza, se AL_0=1 ed AL_1=0 in uscita avremo il secondo operando immutato, se AL_0=1 ed AL_1=1 in uscita avremo il complemento del secondo operando
- S_{alu} = segnale di abilitazione al caricamento sul bus dati dell'output dell'ALU
- R e W per la Memoria: se W=1 ed R=0 si ordina una scrittura, se W=0 ed R=1una lettura, se W=0 ed R=0 rimane invariata, se W=1 ed R=1 una reset

Il funzionamento del registro PC è definito come segue:

- se A_{PC} = 0 allora il contenuto del registro rimane invariato;
- A_{PC} = 1 allora

- Se K_{PC} = 0 allora il PC carica dall'esterno (bus indirizzi)
- Se K_{PC} = 1 allora il PC incrementa il suo contenuto (indirizzo della prossima istruzione)

Il funzionamento del registro IR è definito come segue:

- se A_{IR} = 0 allora il contenuto del registro rimane invariato;
- A_{IR} = 1 allora
  - Se Z_{IR} = 0 allora l'IR carica dall'esterno (bus dati)
  - Se Z_{IR} = 1 allora l'IR si azzera

MICROPROGRAMMI

Ogni istruzione del linguaggio macchina viene decomposto in una sequenza di **microistruzioni** ognuna delle quali è **direttamente eseguibile** dalla parte operativa della CPU.
Ogni sequenza di microistruzioni associate ad una data istruzione macchina è un **microprogramma**.

NOTA: durante l'esecuzione di una istruzione del linguaggio macchina, tale istruzione è memorizza nel registro IR

STORE x
- m1: IR_{4-15} → MAR; : Acc → MDR
- m2: WriteMem (cioè MDR→M[MAR])

LOAD x
- m3: IR_{4-15} → MAR
- m4: ReadMem (cioè M[MAR]→MDR)
- m5: MDR → Acc

ADD x
- m6: Acc → T0 ; IR_{4-15} → MAR
- m7 : ReadMem
- m8 : MDR → T1
- m9: Alu(T0+T1) → Acc

SUB x
- m10: Acc → T0 ; IR_{4-15} → MAR
- m11 : ReadMem
- m12 : MDR → T1
- m13: Alu(T0-T1) → Acc

JUMP x

- m14: $IR_{4\text{-}15}$ → PC

JZ x

If β_{Acc} then

- m15: $IR_{4\text{-}15}$ → PC

L'esecuzione di una microistruzione è determinata dai valori assunti dai segnali di controllo. Nella seguente tabella sono riportati tre esempi (le microistruzioni della STORE):

Segnali di controllo	m1: IR → MAR	m2: Acc → MDR	WRITE
Aacc	0	0	-
Sacc	-	1	-
AIR	0	0	0
SIR	1	-	-
ZIR	0	0	0
AMAR	1	-	0
SMAR	0	-	0
AMDR	-	1	0
SMDR	-	0	0
APC	0	0	0
SPC	0	-	-
KPC	-	-	-
AL_0	-	-	-
AL_1	-	-	-
Salu	-	0	-
R	0	0	0
W	0	0	1

NOTA: I segnali di controllo sono inviati alla Parte Operativa dalla Unità di Controllo

L'ISTRUZIONE FETCH

E' l'istruzione che viene eseguita dalla CPU per portare nell'IR la prossima istruzione del programma che deve essere eseguita (passo 1 del "ciclo della CPU"). Va eseguita all'inizio del programma ed alla fine dell'esecuzione di ogni singola istruzione.

Il microprogramma dell'istruzione FETCH è il seguente:

- m16: PC → MAR
- m17: ReadMem; PC = PC+1;
- m18: MDR → IR

Il risultato del suddetto microprogramma è il trasferimento nell'IR dell'istruzione che si trova in M all'indirizzo contenuto nel PC. Si noti che il PC viene incrementato per "puntare" alla prossima istruzione del programma in M. Tale incremento viene effettuato in parallelo alla Read in quanto non vi è "interferenza" tra le due microistruzioni (non vi è ragione perché una preceda l'altra ed inoltre la loro esecuzione coinvolge componenti diversi dell'architettura).

NOTA: Read e PC = PC+1 costituiscono un'unica microistruzione (la m17)

Assumiamo che il codice operativo di tale istruzione sia 000.

NOTA: per poter "forzare" l'istruzione FETCH al termine di ogni singola istruzione del programma, è necessario porre a zero i tre bit $\mathbf{I_0}$, $\mathbf{I_1}$, $\mathbf{I_2}$ del codice operativo dell'istruzione corrente memorizzata nel registro IR (infatti il codice operativo della FETCH è 000). A tal fine bisogna aggiungere al termine di **ogni** microprogramma una istruzione di azzeramento del registro IR. Ciò si realizza fisicamente mettendo ad 1 il segnale $Z_{IR.}$ Ad esempio

STORE x

- m1: $IR_{4\text{-}15}$ → MAR; m2: Acc → MDR
- m2: WriteMem
- m3: Azzera IR

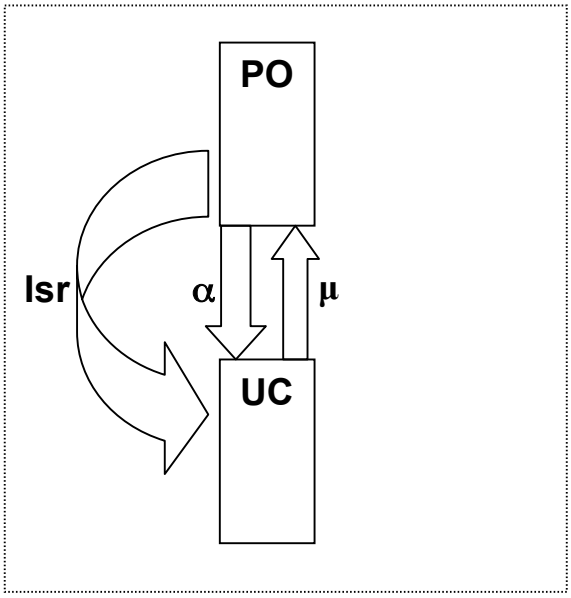
L'UNITA' DI CONTROLLO

E' una rete sequenziale che riceve in ingresso i seguenti segnali di **condizione**:

- I tre bit $\mathbf{I_0}$, $\mathbf{I_1}$, $\mathbf{I_2}$ del codice operativo dell'istruzione corrente memorizzata nel registro IR
- Il segnale $\beta_{\mathbf{Acc}}$

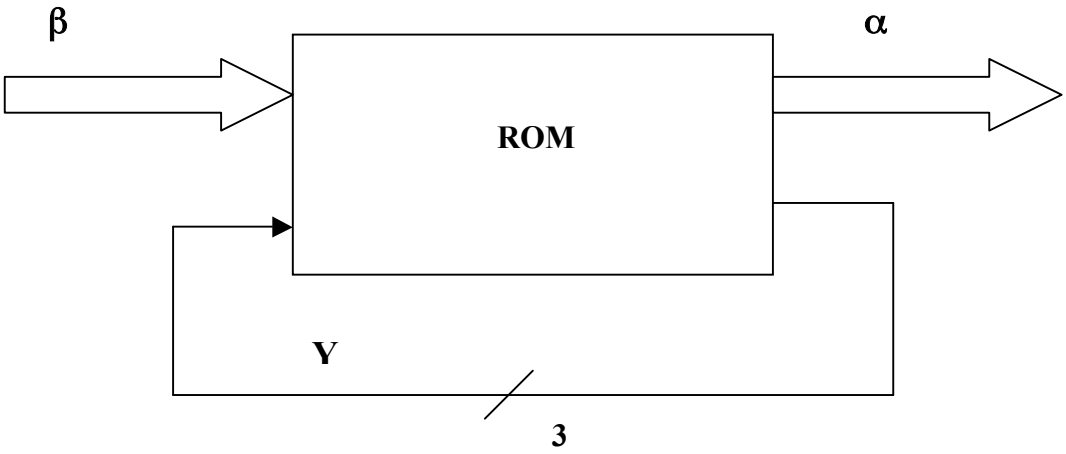
e fornisce in uscita i segnali di controllo della parte operativa (nel nostro caso 19 bit di controllo: A_{acc}, S_{acc} , …., W, R).

Lo schema di interazione tra PO e UC è il seguente:



LEGENDA
PO: PARTE OPERATIVA
UC: PARTE DI CONTROLLO
X: DATI IN INPUT
Z: DATI IN OUTPUT
μ: SEGNALI DI CONTROLLO (COMANDI PER LA PO)
α: SEGNALI CONDIZIONE
Isr: Codice Operativo Istruzione da eseguire (anch'essi segnali di condizione)

La rete sequenziale che implementa la PC ha, nel nostro caso, 2 variabili di anello, 4 variabili d'ingresso e 19 di uscita. La sua parte combinatoria può essere realizzata con una ROM. Il numero di variabili di anello è $log_2 n$, dove n è il massimo numero di micro-istruzioni contenute in un micro-programma



UNITA' DI CONTROLLO

Il comportamento ai morsetti della rete sequenziale Unità di Controllo è il seguente (la mappa è parziale):

	INPUT							OUTPUT			
	I_0	I_1	I_2	β	Y_0	Y_1		Segnali α per la microistruzione	Y'_0	Y'_1	
Fetch	0	0	0	-	0	0		m16	0	1	
	0	0	0	-	0	1		m17	1	0	
	0	0	0	-	1	0		m18	0	0	
Store	0	0	1	-	0	0		m1	0	1	
	0	0	1	-	0	1		m2	1	0	
	0	0	1	-	1	0		Azzera IR	0	0	
JZ	1	0	1	0	0	0		m15	0	1	
	1	0	1	1	0	0		Azzera IR	0	0	
	1	0	1	0	0	1		Azzera IR	0	0	

NOTA: ogni microprogramma termina con una istruzione di azzeramento del registro IR e con un ritorno allo stato 000 della rete sequenziale; in tal modo i valori dell'input della rete Unità di Controllo I_0 I_1 I_2 Y_0 Y_1 sono tutti uguali a zero; ne consegue che la successiva istruzione che viene eseguita è l'istruzione FETCH.