

Elementi di Architettura

Corso di *Fondamenti di Informatica*
(Roberto BASILI)

Classi di Istruzioni

- Istruzioni di assegnazione/modifica
- Istruzioni di controllo
- Istruzioni di I/O

Classi di Istruzioni

- Istruzioni di assegnazione/modifica

Formalizzazione in linguaggio naturale

Es.

- Assegna ad A il valore 2
- $A = 2$ o meglio $A \leftarrow 2$
- Sia 0 il valore iniziale di I
- $I \leftarrow 0$
- $A \leftarrow A + 1$

Classi di Istruzioni

■ Istruzioni di assegnazione/modifica

Es. <Date due variabili A e B>, Scambio di valori tra A e B

1. Scrivi in Temp il valore corrente di A
2. Scrivi in A il valore corrente di B
3. Scrivi in B il valore corrente di Temp

```
1. Temp ← A
2. A ← B
3. B ← Temp
```

Classi di Istruzioni

■ Istruzioni di controllo (1)

Es.

Se (A>B) allora il MASSIMO e' A
altrimenti il MASSIMO e' B

```
Se (A>B) allora
    MASSIMO ← A
altrimenti
    MASSIMO ← B
```

Classi di Istruzioni

■ Istruzioni di controllo (1)

Prevedono:

- Una condizione (a valori booleani)
(i.e. $f : D \rightarrow \{0,1\}$)
- Una modifica della sequenza delle istruzioni

Classi di Istruzioni

■ Istruzioni di Controllo (2)

Es.

1. $R \leftarrow$ resto della divisione di A per B
2. Se $R=0$ allora B è il M.C.D.
3. Altrimenti ($R \neq 0$)
 $A \leftarrow B$;
 $B \leftarrow R$.
4. Torna all'istr.1

Classi di Istruzioni

■ Istruzioni di Controllo (3)

1. $R \leftarrow 1$
2. $MCD \leftarrow 0$
3. Finche' $R \neq 0$ esegui
 - 3.1 $R \leftarrow$ resto di A diviso B
 - 3.2 Se $R=0$ allora
 - 3.2.1 $MCD \leftarrow B$
 - Altrimenti
 - 3.2.2 $A \leftarrow B$
 - 3.2.3 $B \leftarrow R$
4. <Stampa MCD>

Istruzioni di I/O

■ Gestiscono l'ingresso dei dati

(es. Leggi il valore di x,
nel calcolo di una f(x))

■ Provvedono alla trasmissione in uscita dei dati

(es. $y \leq f(x)$)

Stampa il valore di y su schermo)

Istruzioni di I/O

```
■ Es. Determinare il massimo di N numeri letti in
ingresso
■ i ← 0, max ← 0
  Finche' i < N
    <Leggi X>
    Se X > max allora max ← X
    i ← i + 1
  <Stampa max>
```

Istruzioni

```
■ Es. Calcolare la divisione A/B intera
Quoz ← 0
<Leggi A>
<Leggi B>
Finche' B <= A
  A ← A - B
  Quoz ← Quoz + 1
<Stampa Quoz>
```

Elementi di Assembler

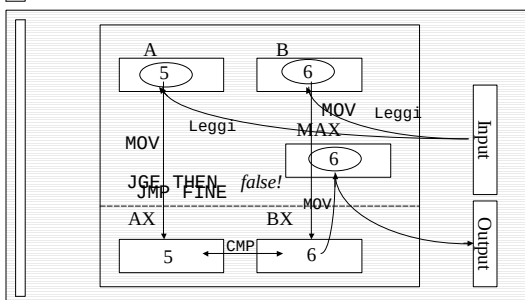
```
■ Un esempio (1)
1. <Leggi A>
2. <Leggi B>
3. 3.1 Se A > B allora il
   3.1.1 MASSIMO e' A
   3.2 altrimenti
   3.2.1 il MASSIMO e' B
4. <Scrivi Massimo>
```

Elementi di Assembler

■ Un esempio (2)

```
- MAX DW ?  
- A DW ?  
- B DW ?  
- <leggi A>  
- <leggi B>  
- MOV AX, A  
- MOV BX, B  
- CMP AX, BX  
- * JGE THEN  
- MOV MAX, BX  
- * JMP FINE  
- THEN: MOV MAX, AX  
- FINE: <scrivi MAX>
```

Elementi di Assembler



Elementi di Assembler

■ Un secondo esempio (1)

```
- Quoz ← 0  
- <Leggi A>  
- <Leggi B>  
- Finche' B ≤ A  
- A ← A - B  
- Quoz ← Quoz + 1  
- <Stampa Quoz>
```

Elementi di Assembler

■ Un secondo esempio (2)

```
-      A   DW  ? ... B   ... DW  ? Q   DW
-      <leggi A>
-      <leggi B>
-      MOV  Q, 0      Q ← 0
-      MOV  AX, A
-  CICLO: SUB  AX, B    AX ← AX - B
-          INC  Q      Q ← Q + 1
-          CMP  B, AX   B ≤ AX
-          JG   END     se B > AX
-          JMP  CICLO
-  END:   <scrivi Q>
```

Elementi di Assembler

■ Struttura di un programma Assembler

Dichiarazioni di variabili
Blocco delle istruzioni

- Sequenza
- Operazioni e operandi

Elementi di Assembler

■ Struttura di un programma assembler

Variabili, (es. A, B, MAX)

Variabili predichiarate (es. AX, BX)

Operazioni del linguaggio Assembler

■ Operatore ADD :

- Frase lecita: ADD dst, src
 - (dove src e dst sono due operandi)
- Semantica: $dst = dst + src$
- Vincoli:
 - l'operando *dst* non può essere specificato come costante oppure come <nome> se <nome> è un nome di costante
 - uno dei due operandi deve essere una variabile predichiarata o una costante

■ Es. ADD A, BX ADD AX, 1 ADD A, 3

Operazioni del linguaggio Assembler

■ Operatore SUB:

- Frase lecita: SUB dst, src
 - (dove src e dst sono due operandi)
- Semantica: $dst = dst - src$
- Vincoli:
 - l'operando *dst* non può essere specificato come costante oppure come <nome> se <nome> è un nome di costante
 - uno dei due operandi deve essere una variabile predichiarata o una costante

■ Es. SUB A, BX SUB AX, 1 SUB A, 3

Operazioni del linguaggio Assembler

■ Operatore MUL:

- Frase lecita: MUL dst, src
 - (dove src e dst sono due operandi)
- Semantica: $dst = dst * src$
- Vincoli:
 - l'operando *dst* non può essere specificato come costante oppure come <nome> se <nome> è un nome di costante
 - uno dei due operandi deve essere una variabile predichiarata o una costante

■ Es. MUL A, BX MUL AX, 1 MUL A, 3

Operazioni del linguaggio Assembler

■ Operatore DIV:

- Frase lecita: `DIV dst, src`
 - (dove `src` e `dst` sono due operandi)
 - Semantica: $dst = dst / src$
 - Vincoli:
 - l'operando `dst` non può essere specificato come costante oppure come `<nome>` se `<nome>` è un nome di costante
 - uno dei due operandi deve essere una variabile predichiarata o una costante
- Es. `DIV A, BX` `DIV AX, 1` `DIV A, 3`

Operazioni del linguaggio Assembler

■ Operatore INC:

- Frase lecita: `INC dst`
 - (dove `dst` e' una variabile)
 - Semantica: $dst = dst + 1$
 - Vincoli:
 - l'operando `dst` non può essere specificato come costante oppure come `<nome>` se `<nome>` è un nome di costante
- Es. `INC A` `INC AX`

Operazioni del linguaggio Assembler

■ Operatore DEC:

- Frase lecita: `DEC dst, src`
 - (dove `dst` e' una variabile)
 - Semantica: $dst = dst - 1$
 - Vincoli:
 - l'operando `dst` non può essere specificato come costante oppure come `<nome>` se `<nome>` è un nome di costante
- Es. `DEC A` `DEC AX`

Operazioni del linguaggio Assembler

■ Operatore CMP:

- Frase lecita: **CMP dst, src**
 - (dove src e dst sono due operandi)
- Semantica: confronta (e memorizza l'esito del confronto) *dst* e *src*
- Vincoli:
 - uno dei due operandi deve essere una variabile predichiarata o una costante

■ Es. **CMP A, BX CMP AX, 1 CMP A, 3**

Istruzione di Salto

■ Operatore JMP:

- Frase lecita: **JMP <etichetta>** (<etichetta> e' l'unico operando)
- Semantica: Salta alla istruzione la cui etichetta e' <etichetta>
- Vincoli: l'istruzione precedente deve essere di confronto tra due variabili o valori

(assegna a PC il valore di puntatore alla istruzione nel cui campo etichetta compare <etichetta>)

Istruzione di Salto

■ Salto Condizionato :

- Frase lecita: **J<COND> <etichetta>** (<etichetta> e' l'unico operando)
- Semantica: Se <Cond> e' vera salta alla istruzione la cui etichetta e' <etichetta>
- Vincoli: l'istruzione precedente deve essere di confronto tra due variabili o valori

Istruzione di Salto

■ Salto Condizionato :

- Frase lecita: **J<COND> <etichetta>** (<etichetta> e' l'unico operando)

JMP nessuna
JG <cond> = il confronto ha dato esito >
JGE <cond> = il confronto ha dato esito ≥
JL <cond> = il confronto ha dato esito <
JLE <cond> = il confronto ha dato esito ≤
JEQ <cond> = il confronto ha dato esito =
JNZ <cond> = il confronto ha dato esito ≠

... verso l'architettura

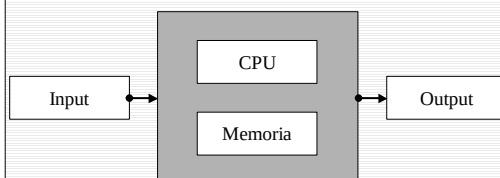
■ Cosa serve per “calcolare” (*eseguire*) un programma Assembler?

- Rappresentare (i.e. Memorizzare) i dati
- Localizzare i dati
- Rappresentare Istruzioni e Operandi corrispondenti
- Controllare la sequenza delle istruzioni



Architettura di un Elaboratore

■ Von Neumann (1950)



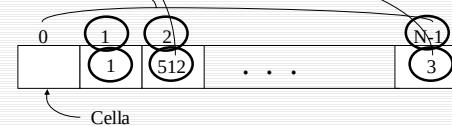
Organizzazione della memoria

■ Unita' atomica, *bit*

■ Cella (o *Parola*)

■ Valori

■ Indirizzi



Organizzazione della memoria

■ Proprietà

- Dimensione, o *Spazio di Indirizzamento*

■ i.e. **numero delle celle N**

- *Tempo d'accesso*

■ i.e. intervallo di trasferimento da/a memoria

Organizzazione della memoria

■ Bit e Parole

	m-1	m-2	m-3	3	2	1	0
0000000				...			
0000001				...			
0000010				...			
...							
0010011	0	0	0	...	1	0	1
...							
1111111				...			

Organizzazione della memoria

- Per una memoria di dimensione K (cioè K celle disponibili) ho bisogno di un numero n di bit di indirizzamento tale che

$$2^n \geq K$$

- Per evitare ridondanze K è scelto tale che

$$2^n = K$$

- 1 **byte** = 8 **bit**
- 1 **kilobyte** = 1 **Kbyte** = 2^{10} bytes = 1,024 bytes
- 1 **megabyte** = 1 **Mbyte** = 2^{20} bytes 1,000,000 bytes

Operazioni della Memoria

- *Lettura* di un dato (*fetch*)
da Memoria a CPU
- *Scrittura* di un dato (*store*)
da CPU a Memoria
- Canali di Comunicazione (*bus*)
 - Dati
 - Indirizzi
 - Controllo (Lettura/Scrittura)

Operazioni della Memoria

- *Nella Lettura*
La CPU mette a disposizione un **indirizzo**
- *Nella Scrittura*
La CPU mette a disposizione un **indirizzo** ed un **valore**

Operazioni della Memoria

■ Due registri gestiscono il traffico:

– Memory Address Register (MAR)

- contiene l'indirizzo della cella di memoria da cui eseguire la lettura/scrittura

– Memory Buffer Register (MBR)

- Scrittura: contiene il valore da memorizzare

$M(MAR) \leftarrow MBR$

- Lettura: riceve il valore della cella di indirizzo MAR

$M(MAR) \rightarrow MBR$

Gerarchia della Memoria

Registri

Veloci, Costosi

RAM (Random Access Memory)

Meno veloce, meno costosa

Memoria Secondaria

Lenta, bassissimi costi

Cache

tra RAM e registri

Gerarchia della Memoria

Registri

Processori	Numero registri	Dimensione
DEC VAX	16	32
Intel 8086	8	16
Intel 80386	8	32
Motorola 68000	16	32
Sun Sparc	128	32

Il Processore Centrale (CPU)

... cosa deve fare?

■ Ciclo Operativo (*Instruction Cycle*)

- Localizza la istruzione successiva *I*
- Carica l'istruzione *I*
- Decodifica *I*
- Esegue *I*

Il Processore Centrale (CPU)

L'Algoritmo:

- Finché ($I \neq \text{HALT}$) {
 - Localizza *I*
 - Carica *I*
 - Decodifica *I*
 - Esegui *I*}

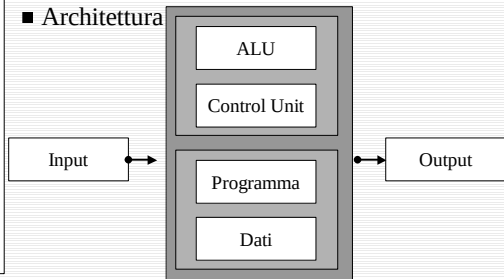
Il Processore Centrale (CPU)

■ Elementi Costitutivi

- ALU (Arithmetic-Logic Unit)
 - Opera sui dati
 - Operazioni aritmetiche, relazioni logiche, trasmissione
- Control Unit (CU o UC)
 - Gestisce il ciclo operativo
 - Eventuali condizioni di errore (per es, overflow)
 - Interruzioni asincrone

Il Processore Centrale (CPU)

■ Architettura



Il Processore Centrale (CPU)

■ Localizzazione delle Istruzioni

- Program Counter (PC)
 - E' il registro in cui in ogni istante e' memorizzato l'indirizzo della prossima istruzione
- Instruction Register (IR)
 - E' il registro che contiene la istruzione in fase di esecuzione

Il Processore Centrale (CPU)

■ Ciclo di fetch della istruzione

- $MAR \leftarrow PC$
- $MBR \leftarrow M(MAR)$ % lettura istruzione
- $IR \leftarrow MBR$
- $PC \leftarrow PC + s$ % s e' la taglia della istruzione

Il Processore Centrale (CPU)

■ Esecuzione delle Istruzioni

– Decodifica Operazione

- Le operazioni possibili corrispondono a rappresentazioni, dette *codici operativi*
- Ogni operazione determina anche il tipo dei suoi operandi (Operation Mode)
- Gli operandi hanno una loro rappresentazione
 - Costanti (valori diretti)
 - Registri (accumulatori)
 - Indirizzi in memoria

Il Processore Centrale (CPU)

■ Esecuzione delle Istruzioni

– Decodifica Operazione

- Una istruzione occupa lo spazio necessario a codificare tutte le seguenti informazioni
 - Codice Operativo + Modo
 - Valori diretti *
 - Registro/i associato alla istruzione *
 - Indirizzo associato alla istruzione *

* sta per opzionale cioe' dipendente da Codice operativo e Modo

Il Processore Centrale (CPU)

■ Formato delle Istruzioni

Cod Operativo

ADD AX, B

--	--	--	--

Registro

Indirizzo

Cod Operativo

MOV AX, BX

--	--	--	--	--

Registro Registro

Il Processore Centrale (CPU)

■ Esecuzione (es. ADD AX, A)

<CO-MO, R1, Ind>

- Richiede due accessi in memoria
- Fetch istruzione (come prima)

- Fetch operando

- $MAR \leftarrow Ind$
- $MBR \leftarrow M(MAR)$ % lettura operando
- esegui operazione CO su MBR e R1
il risultato e' memorizzato in R1!

Il Processore Centrale (CPU)

■ Operazioni di JUMP

Cod Operativo

JUMP FINE

--	--	--	--

Registro

Indirizzo

L'Indirizzo fornito dalla istruzione e' quello relativo a **FINE**

Il Processore Centrale (CPU)

■ Esecuzione (es. JMP FINE)

<CO-MO, -, Ind>

- Ind corrisponde a FINE
- CO-MO corrisponde a JMP

- Esecuzione

- $PC \leftarrow Ind$
il risultato e' che cambia la successiva istruzione da eseguire!

Linguaggio Macchina

Il linguaggio basato sulle triple
<C0-M0, R1, R2/Ind>
e' detto **linguaggio macchina**

Macchine CISC

Macchine RISC

Linguaggio Macchina

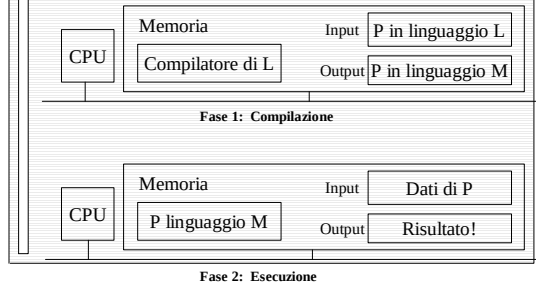
Il linguaggio macchina *corrisponde* a quello
Assembler
Cod. Oper. => Operazioni Assembler
Registri => Variabili Predichiarate
Indirizzi => Nomi Variabili
Indirizzi => Etichette

Linguaggio Macchina

- Il procedimento risolutivo si codifica nel linguaggio a piu' alto livello
- Il processo di traduzione verso il linguaggio macchina e' detto **compilazione**.
- Nel caso (semplice) dell'Assembler esso e' detto Assemblaggio

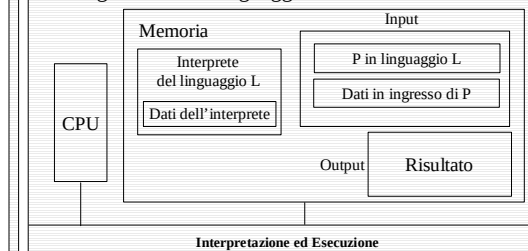
Compilazione e Macchina Virtuale

■ Programma P in linguaggio ad alto livello L



Compilazione e Macchina Virtuale

■ Programma P in linguaggio ad alto livello L



Interpreti vs. Compilatori

■ Caratteristiche

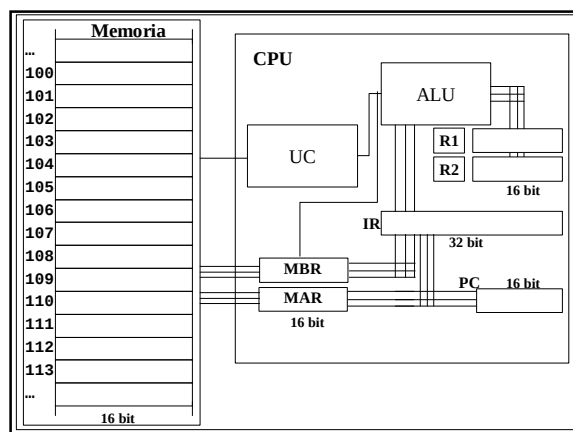
- Spazio di memoria
- Tecniche di Ottimizzazione
- Sicurezza

■ Nuove Tendenze

- Java
- Network Computing

Elementi di Architettura

- Una macchina semplificata
- ... al lavoro



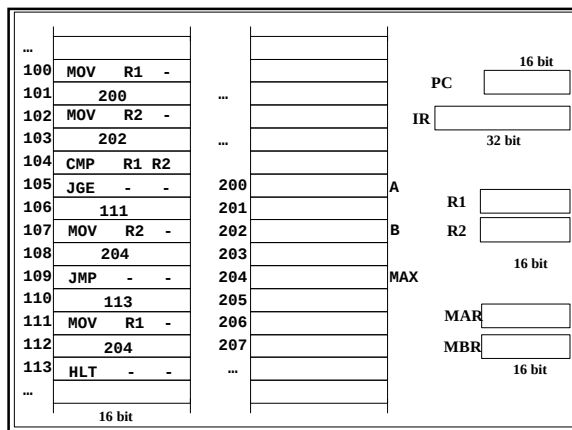
Linguaggio Macchina

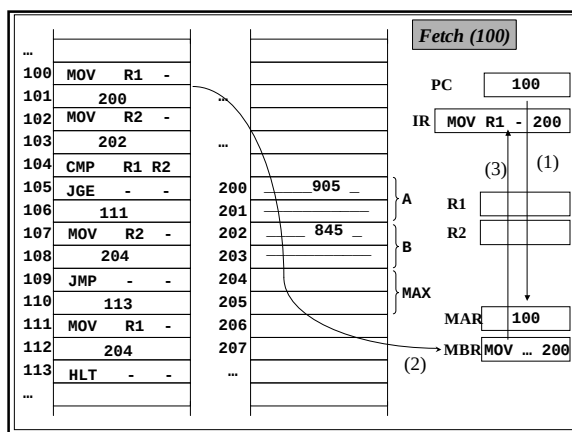
- Esempio: Massimo tra due numeri in Assembler

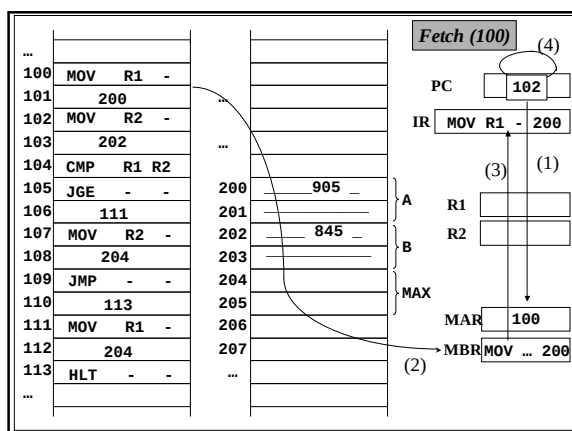
```
MAX DW ?  
A DW ?  
B DW ?
```

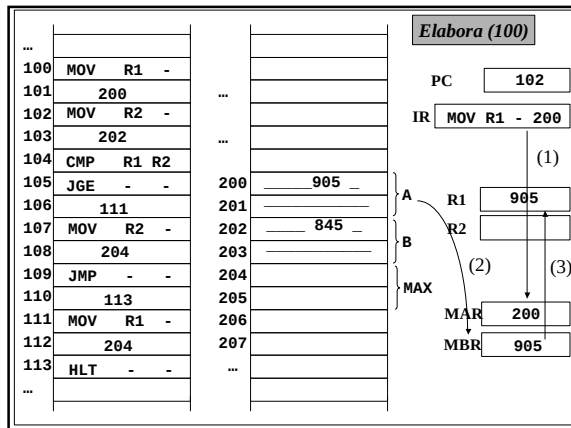
...

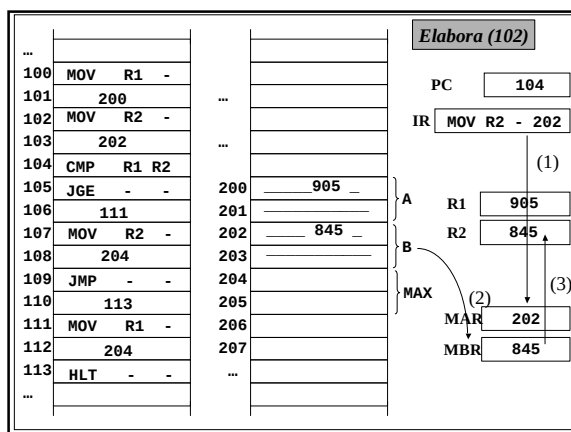
```
MOV AX, A  
MOV BX, B  
CMP AX, BX  
JGE THEN  
MOV MAX, BX  
JMP FINE  
THEN MOV MAX, AX  
FINE HLT
```

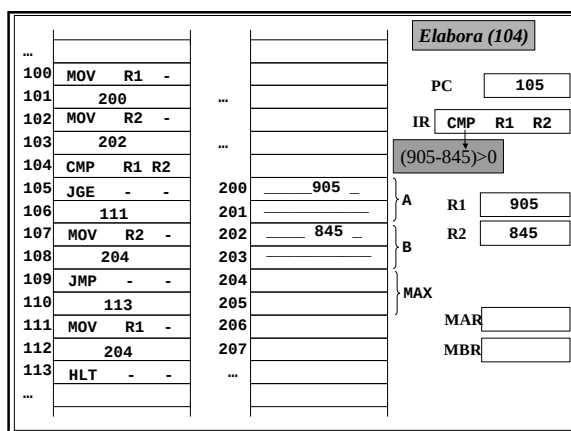








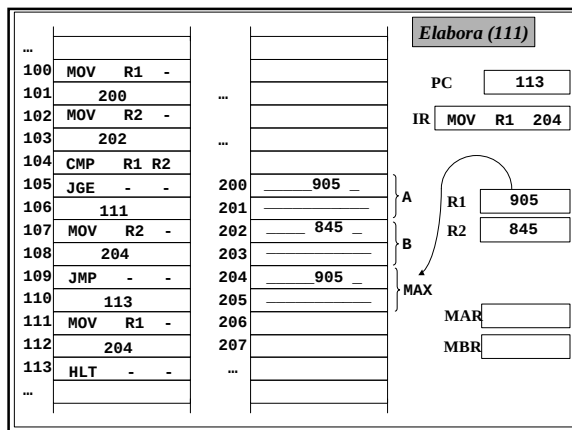




Elabora (105)			
...			
100	MOV R1 -		PC 107
101	200	...	
102	MOV R2 -		IR JGE - 111
103	202	...	
104	CMP R1 R2		(905-845)>0
105	JGE - -	200	905
106	111	201	
107	MOV R2 -	202	845
108	204	203	
109	JMP - -	204	
110	113	205	
111	MOV R1 -	206	
112	204	207	
113	HLT - -	...	
...			

Elabora (105)			
...			
100	MOV R1 -		PC 111
101	200	...	
102	MOV R2 -		IR JGE - 111
103	202	...	
104	CMP R1 R2		(905-845)>0
105	JGE - -	200	905
106	111	201	
107	MOV R2 -	202	845
108	204	203	
109	JMP - -	204	
110	113	205	
111	MOV R1 -	206	
112	204	207	
113	HLT - -	...	
...			

Fine Fetch (111)			
...			
100	MOV R1 -		PC 113
101	200	...	
102	MOV R2 -		IR MOV R1 204
103	202	...	
104	CMP R1 R2		
105	JGE - -	200	905
106	111	201	
107	MOV R2 -	202	845
108	204	203	
109	JMP - -	204	
110	113	205	
111	MOV R1 -	206	
112	204	207	
113	HLT - -	...	
...			



Architetture

■ Tecnologie Avanzate

- Pipelining
- RISC machines
- CISC machines
- Multiprocessori
