Architetture di microprocessori e tecniche di controllo

Modulo di Calcolatori Elettronici

Il problema

Dato un data_path di un processore ed un'istruzione da eseguire:

Come è possibile inviare la sequenza giusta dei segnali alle diverse componenti del data_path per eseguire le operazioni richieste ???

- **Logica cablata**
- **Logica microprogrammata**

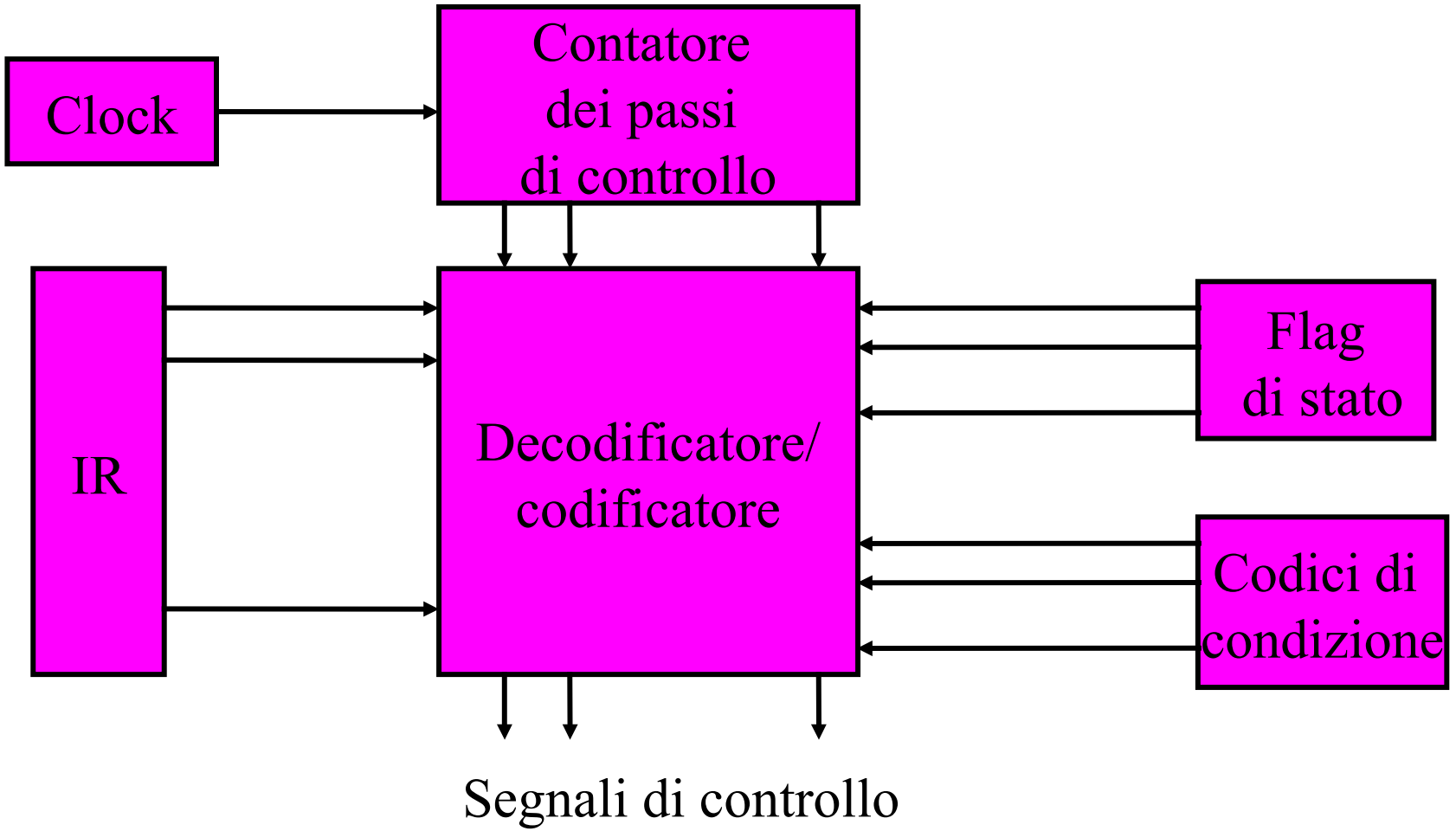
Argomenti

- Processori a logica cablata
- Processori microprogrammati
  - Microistruzioni e microprogrammi
  - Esempio di architettura microprogrammata
- Architetture CISC
- Architetture RISC
  - Pipeline
  - Architetture superscalari

Controllo cablato

- È detto anche hardwired
- La sezione di controllo si basa su una macchina a stati finiti
  - Progettata con le tecniche classiche del progetto di circuiti digitali
  - Spesso con il supporto di strumenti di progettazione integrati (linguaggi/simulatori di hardware)

Controllo Cablato

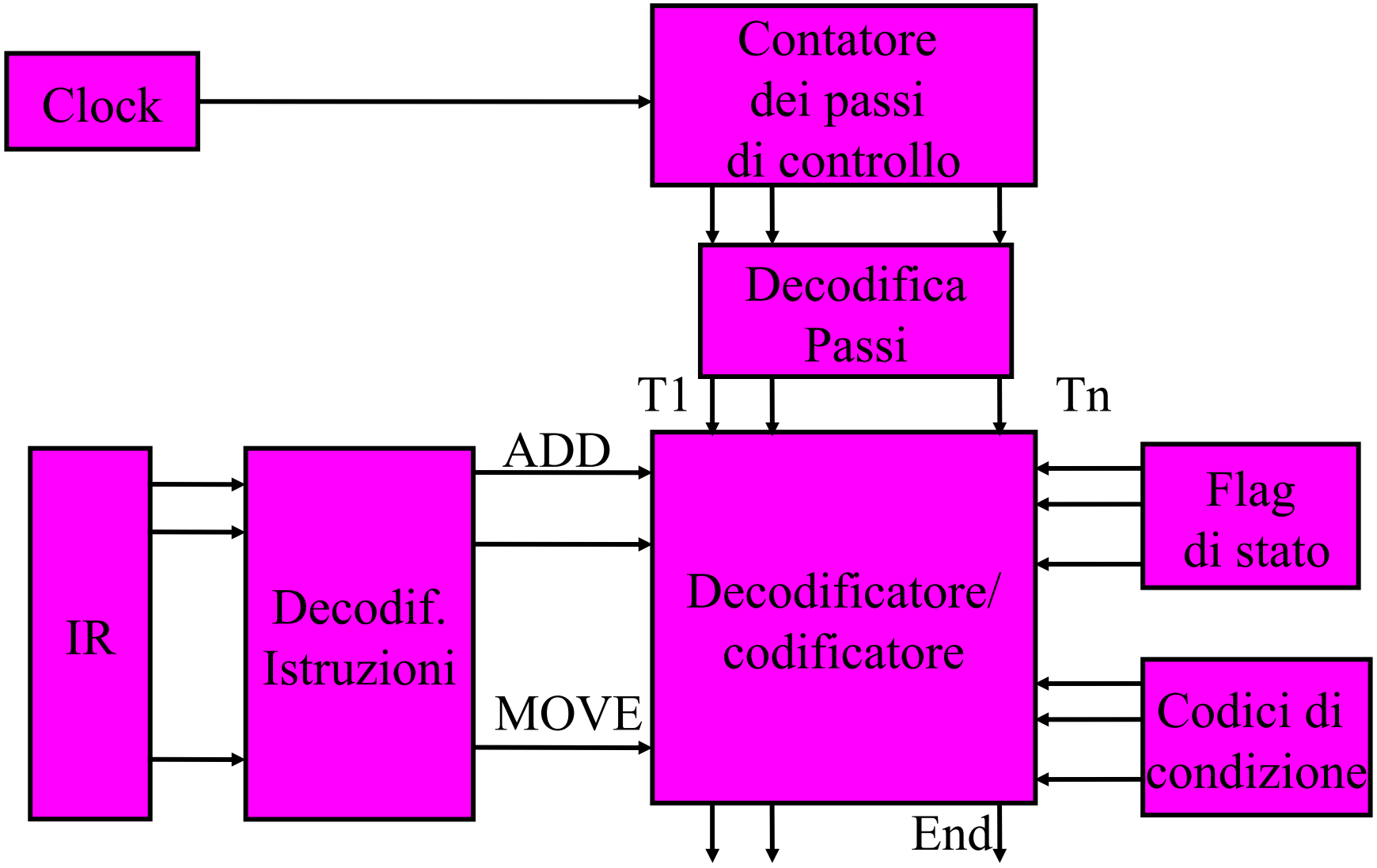


Componenti

- Il clock permette di incrementare il contatore
- Il contatore scandisce i passi necessari all'esecuzione di un'istruzione
- IR contiene l'istruzione da eseguire
- I codici di condizione ed i Flag di stato rappresentano i segnali che danno informazioni riguardo lo stato del processore (registro di stato, segnale di lettura effettuata, memoria pronta…)

I segnali di controllo

- Il contatore scandisce le fasi di esecuzione di un istruzione
- Istruzioni diverse possono durare un numero diverso di fasi
- Il valore dei segnali di controllo è diverso per ogni fase di esecuzione dell'istruzione


Clock
Contatore
dei passi
di controllo
Decodifica
Passi
T1
Tn
IR
Decodif.
Istruzioni
ADD
MOVE
Decodificatore/
codificatore
Flag
di stato
Codici di
condizione
End
Segnali di controllo

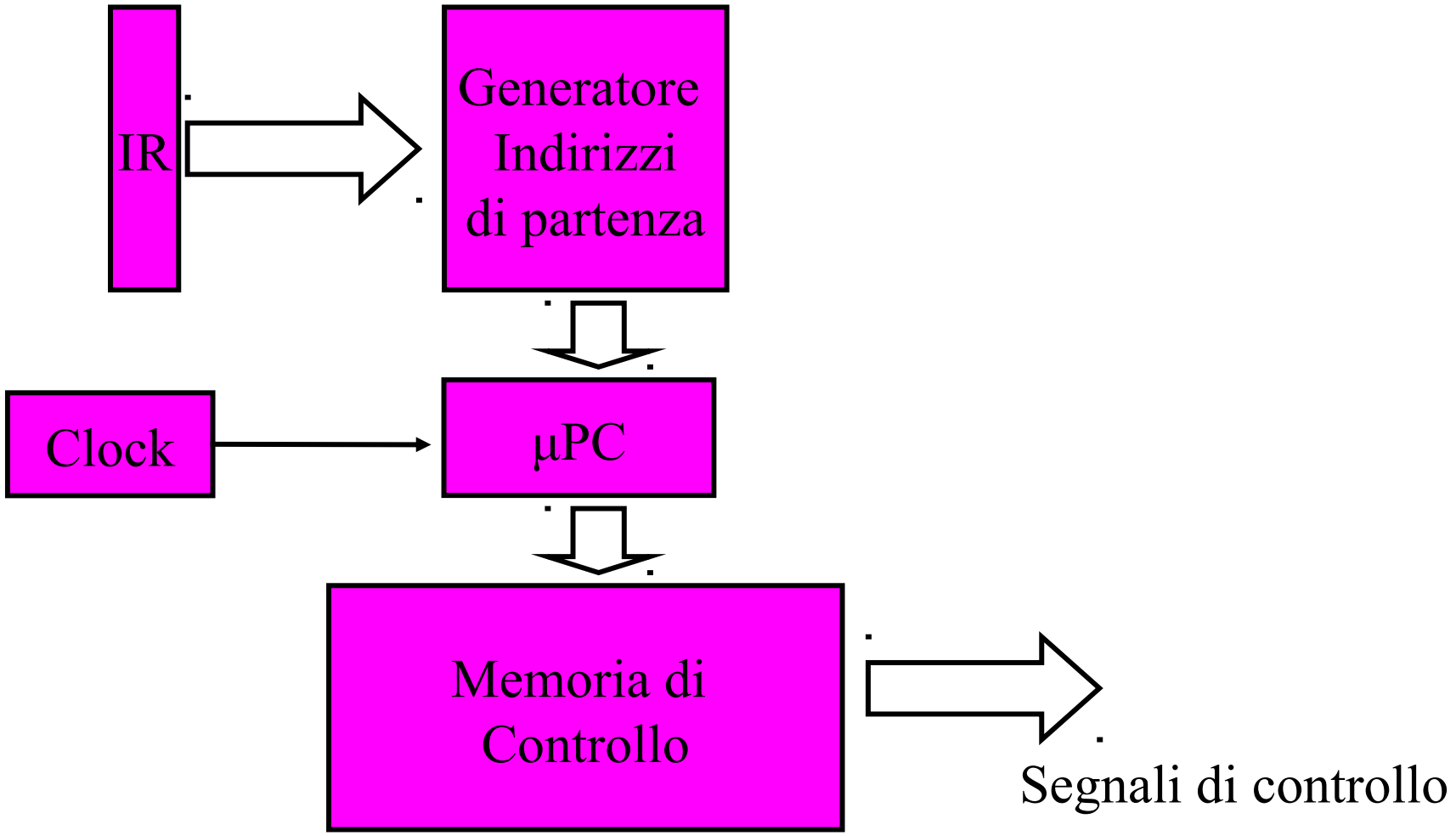
Controllo microprogrammato

- È detto anche microcoded
- La sezione di controllo si basa su un microprogramma
  - Scritto in un linguaggio simile a quello assemblativo
  - Tradotto in codice binario
  - Memorizzato su una ROM (Control Store)

Processori microprogrammati

- Istruzioni macchina “complesse”
  - L’esecuzione di un’istruzione macchina richiede generalmente un data path complesso cioè l’espletamento di più operazioni di trasferimento elementari (microperazioni)
  - Il codice operativo macchina non riesce a contenere tutte le informazioni di tempificazione ed abilitazione dei circuiti e dei registri coinvolti

Schema di Principio



Componenti

- IR contiene prossima istruzione da eseguire
- Generatore degli indirizzi di partenza inizializza il μPC
- Il μPC indirizza la memoria che contiene le microistruzioni
- Le uscite della memoria sono i segnali di abilitazione per il data-path

Considerazioni

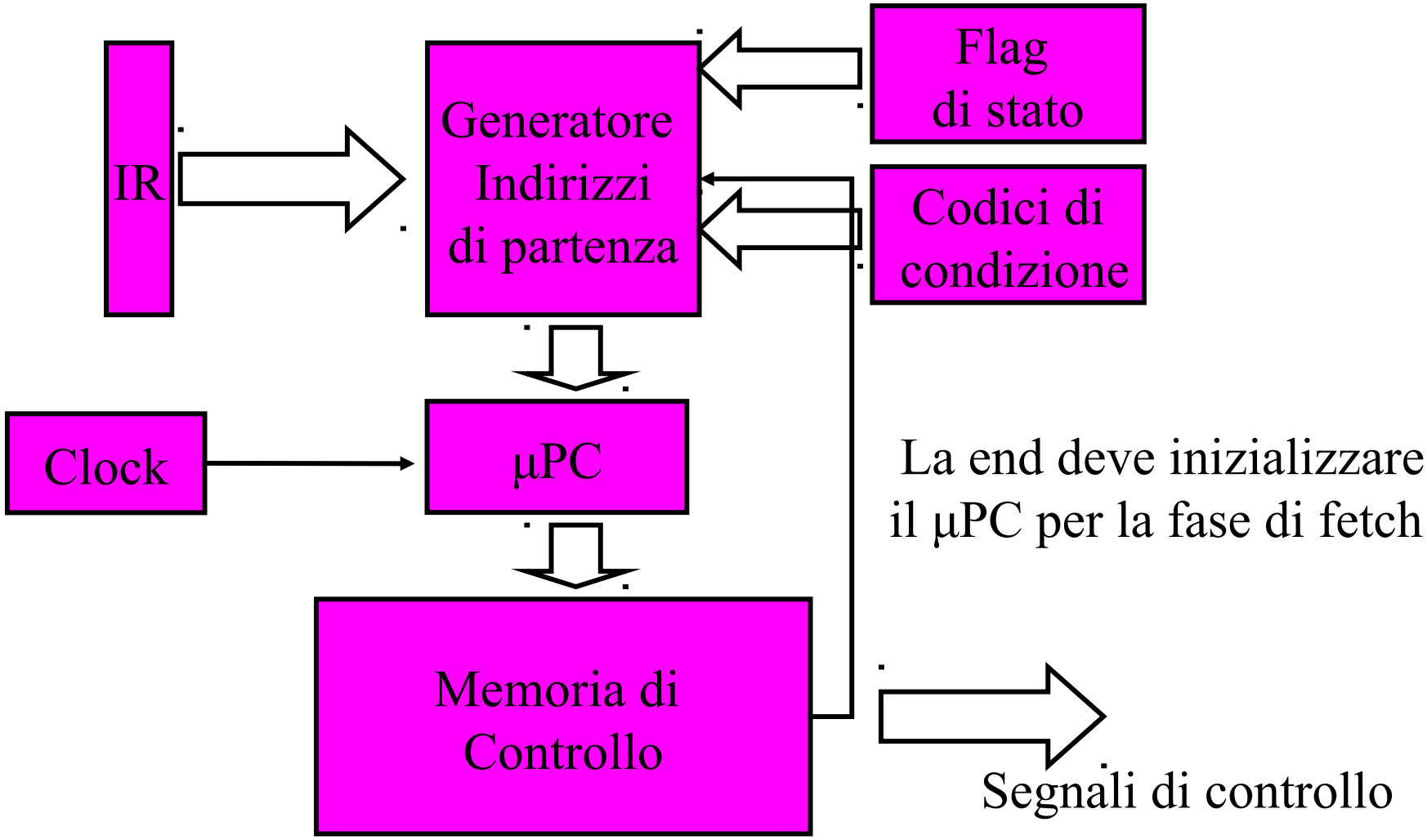
- Una parola della micro-memoria è detta Control Word (CW) e rappresenta una microistruzione
- Occorre inserire nella memoria la sequenza di micro-istruzioni per ogni istruzione assembler
- Occorre inserire anche la sequenza di micro-istruzioni per l'operazione di fetch

I salti

- Per alcune istruzioni assembler la sequenza delle microistruzioni da eseguire dipende da:
  - Codici di condizione
  - Flag di stato

  → Occorre tener conto anche di questi per inizializzare il μ PC

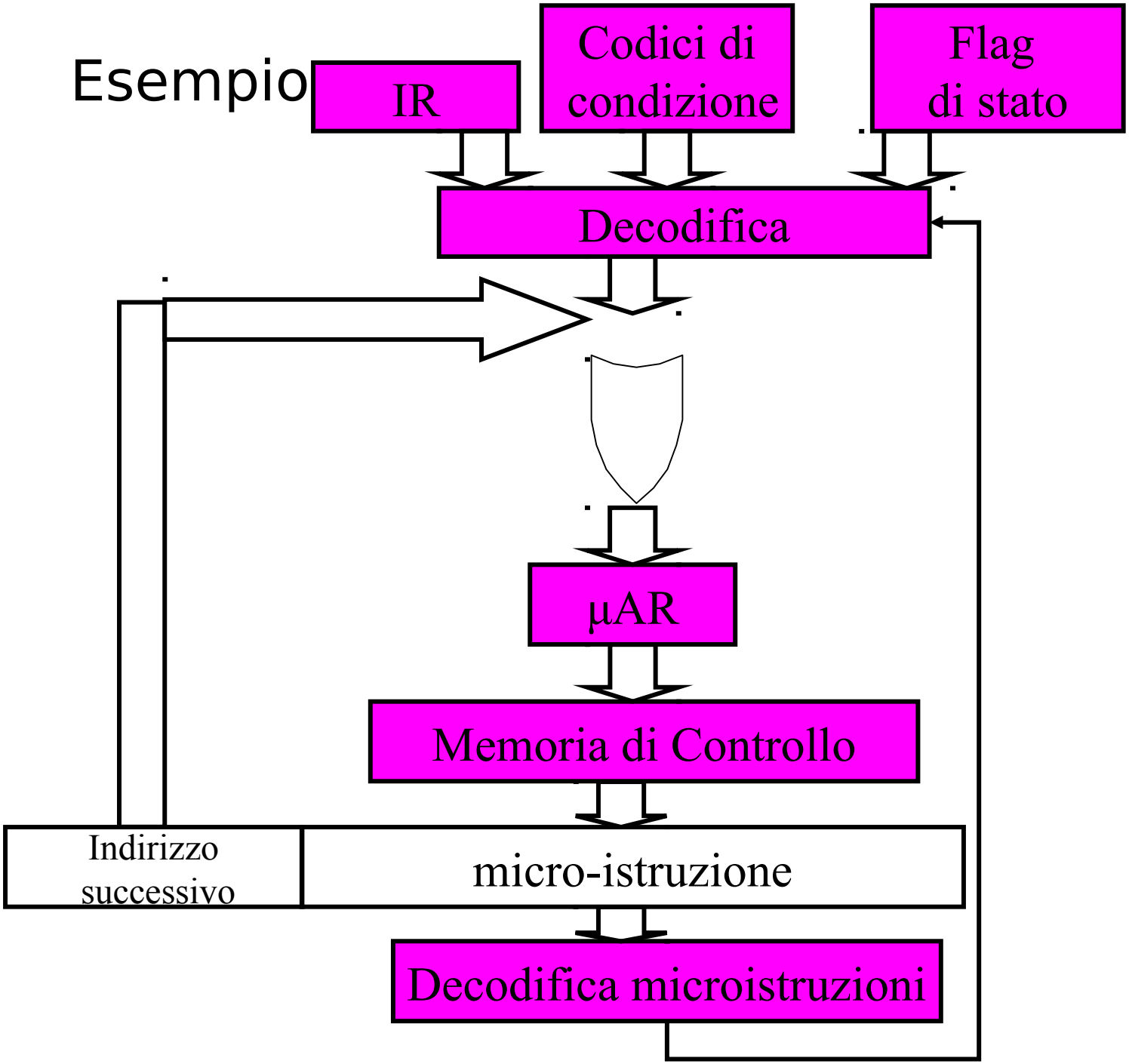
Codici di salto



Controllo flusso

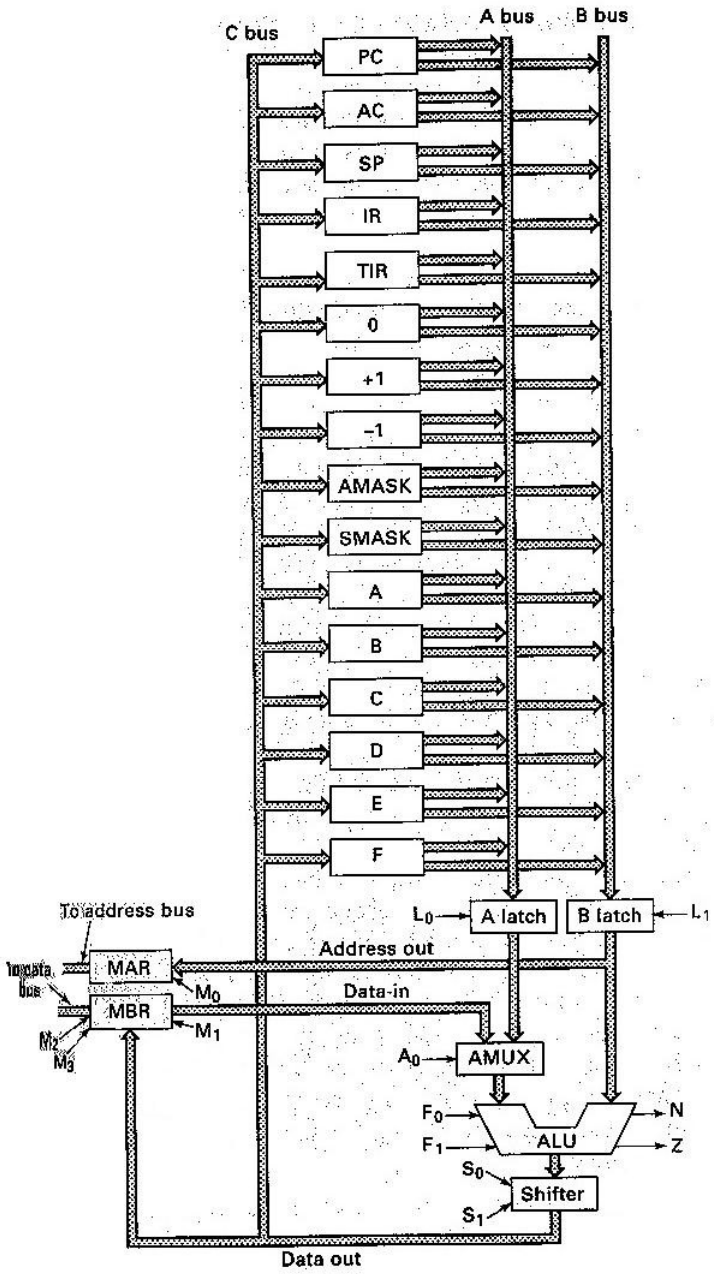
- Memorizzare una nuova sequenza di microistruzioni per ogni istruzione assembler richiede molto spazio di memoria
- La soluzione è prevedere anche microistruzioni di salto
- Quello che si fa è indicare l'ordine delle microistruzioni nella microistruzione stessa

Esempio


IR
Codici di condizione
Flag di stato
Decodifica
µAR
Memoria di Controllo
Indirizzo successivo
micro-istruzione
Decodifica microistruzioni

Esempio di architettura microprogrammata – Data Path

- Data Path
  - **Registri generali**
    - **A, B, C, D, E, F**
  - **Registri speciali**
  - **Bus interni**
    - **A, B, C**



Esempio di architettura microprogrammata - Istruction set

- Livello Assembly

Binary	Mnemonic	Instruction	Meaning
0000xxxxxxxxxxxx	LODD	Load direct	$ac := m[x]$
0001xxxxxxxxxxxx	STOD	Store direct	$m[x] := ac$
0010xxxxxxxxxxxx	ADDD	Add direct	$ac := ac + m[x]$
0011xxxxxxxxxxxx	SUBD	Subtract direct	$ac := ac - m[x]$
0100xxxxxxxxxxxx	JPOS	Jump positive	**if** $ac \geq 0$ **then** $pc := x$
0101xxxxxxxxxxxx	JZER	Jump zero	**if** $ac = 0$ **then** $pc := x$
0110xxxxxxxxxxxx	JUMP	Jump	$pc := x$
0111xxxxxxxxxxxx	LOCO	Load constant	$ac := x\ (0 \leq x \leq 4095)$
1000xxxxxxxxxxxx	LODL	Load local	$ac := m[sp + x]$
1001xxxxxxxxxxxx	STOL	Store local	$m[x + sp] := ac$
1010xxxxxxxxxxxx	ADDL	Add local	$ac := ac + m[sp + x]$
1011xxxxxxxxxxxx	SUBL	Subtract local	$ac := ac - m[sp + x]$
1100xxxxxxxxxxxx	JNEG	Jump negative	**if** $ac < 0$ **then** $pc := x$
1101xxxxxxxxxxxx	JNZE	Jump nonzero	**if** $ac \neq 0$ **then** $pc := x$
1110xxxxxxxxxxxx	CALL	Call procedure	$sp := sp - 1$; $m[sp] := pc$; $pc := x$
1111000000000000	PSHI	Push indirect	$sp := sp - 1$; $m[sp] := m[ac]$
1111001000000000	POPI	Pop indirect	$m[ac] := m[sp]$; $sp := sp + 1$
1111010000000000	PUSH	Push onto stack	$sp := sp - 1$; $m[sp] := ac$
1111011000000000	POP	Pop from stack	$ac := m[sp]$; $sp := sp + 1$
1111100000000000	RETN	Return	$pc := m[sp]$; $sp := sp + 1$
1111101000000000	SWAP	Swap ac, sp	$tmp := ac$; $ac := sp$; $sp := tmp$
11111100yyyyyyyy	INSP	Increment sp	$sp := sp + y\ (0 \leq y \leq 255)$
11111110yyyyyyyy	DESP	Decrement sp	$sp := sp - y\ (0 \leq y \leq 255)$

xxxxxxxxxxx is a 12-bit machine address; in column 4 it is called x.
yyyyyyyy is an 8-bit constant; in column 4 it is called y.

Fig. 4-14. The Mac-1 instruction set.

Esempio di architettura microprogrammata – Esecuzione di un codice operativo

- ADDD X ** AC:=AC + M[x] **

- Fetch
  - MAR:=PC
  - Abilita l'operazione READ in memoria
  - IR:=MBR
  - PC:=PC+1
  - Decodifica istruzione

Esempio di architettura microprogrammata – Esecuzione di un codice operativo (2)

- Operand Assembly
  - MAR:=IR.X
  - Abilita l'operazione READ in memoria
- Execute
  - AC:=AC+MBR

Microprogramma e microistruzioni

- Microistruzioni
  - Specifica una microperazione
    - Calcolo o Trasferimento elementare
  - Ingloba le informazioni di tempificazione e di abilitazione dei circuiti e dei registri coinvolti nella microperazione

- Microprogramma
  - Sequenza ordinata di microistruzioni che esegue le istruzione macchina

Supporto hardware

- Control storage (logica firmware)
  - Memoria ROM per la memorizzazione dei microprogrammi
- Micro Istruction Register
  - Registro per la memorizzazione della microistruzione corrente
- Micro Program Counter
  - Registro per la memorizzazione dell'indirizzo della prossima microistruzione da eseguire

Esempio di architettura microprogrammata

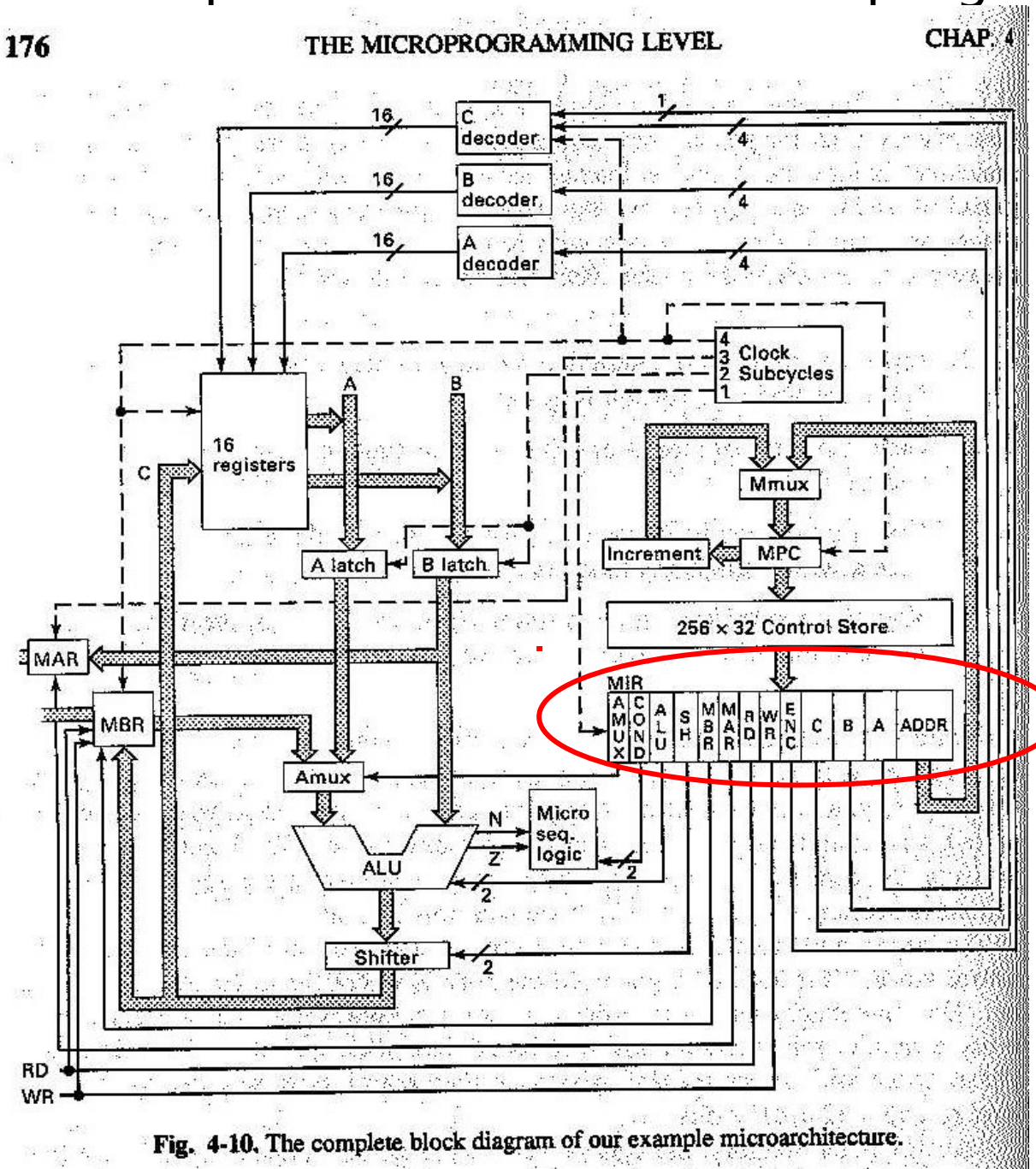


Fig. 4-10. The complete block diagram of our example microarchitecture.

Micro Istruction Register

Esempio di architettura microprogrammata

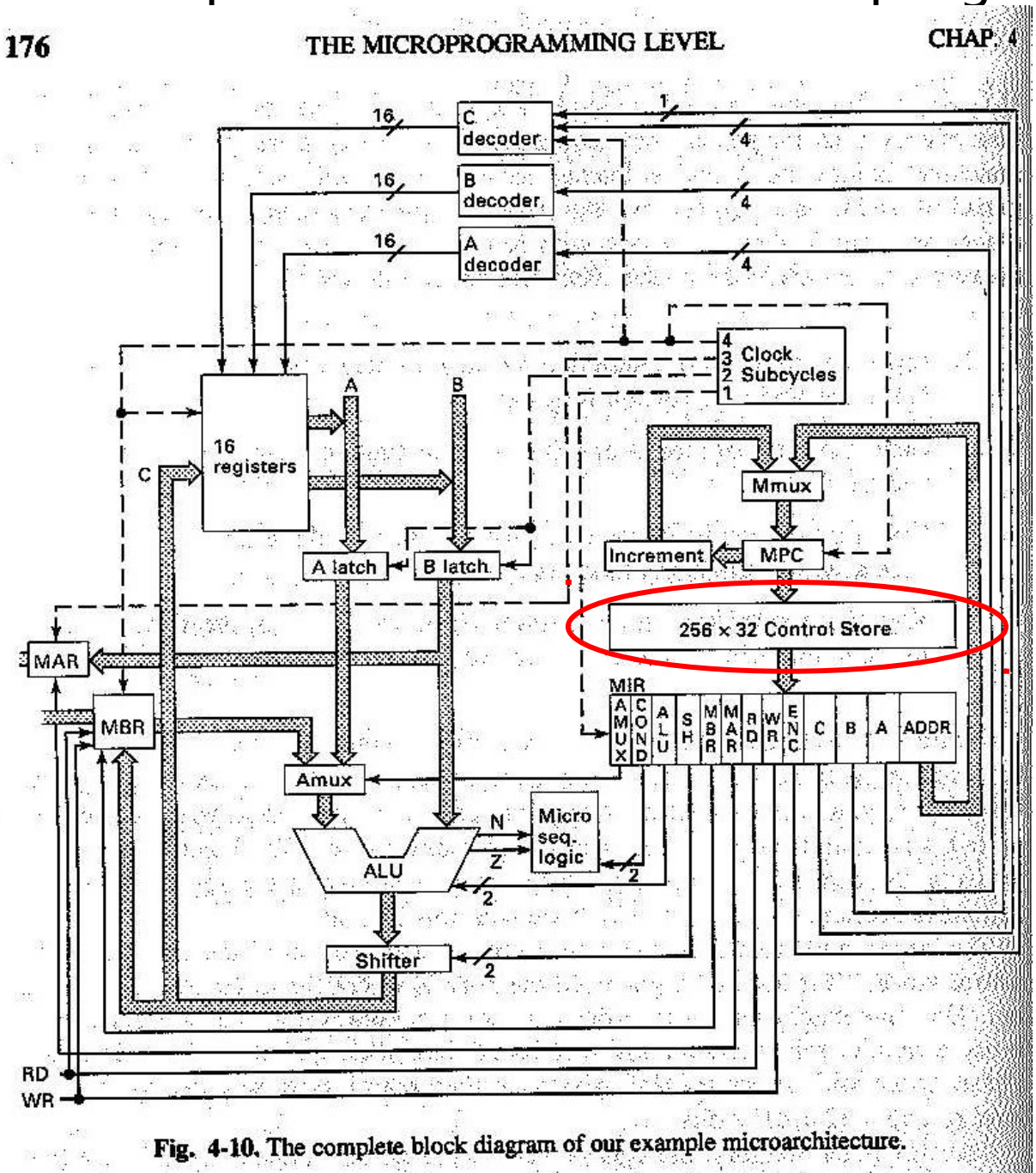
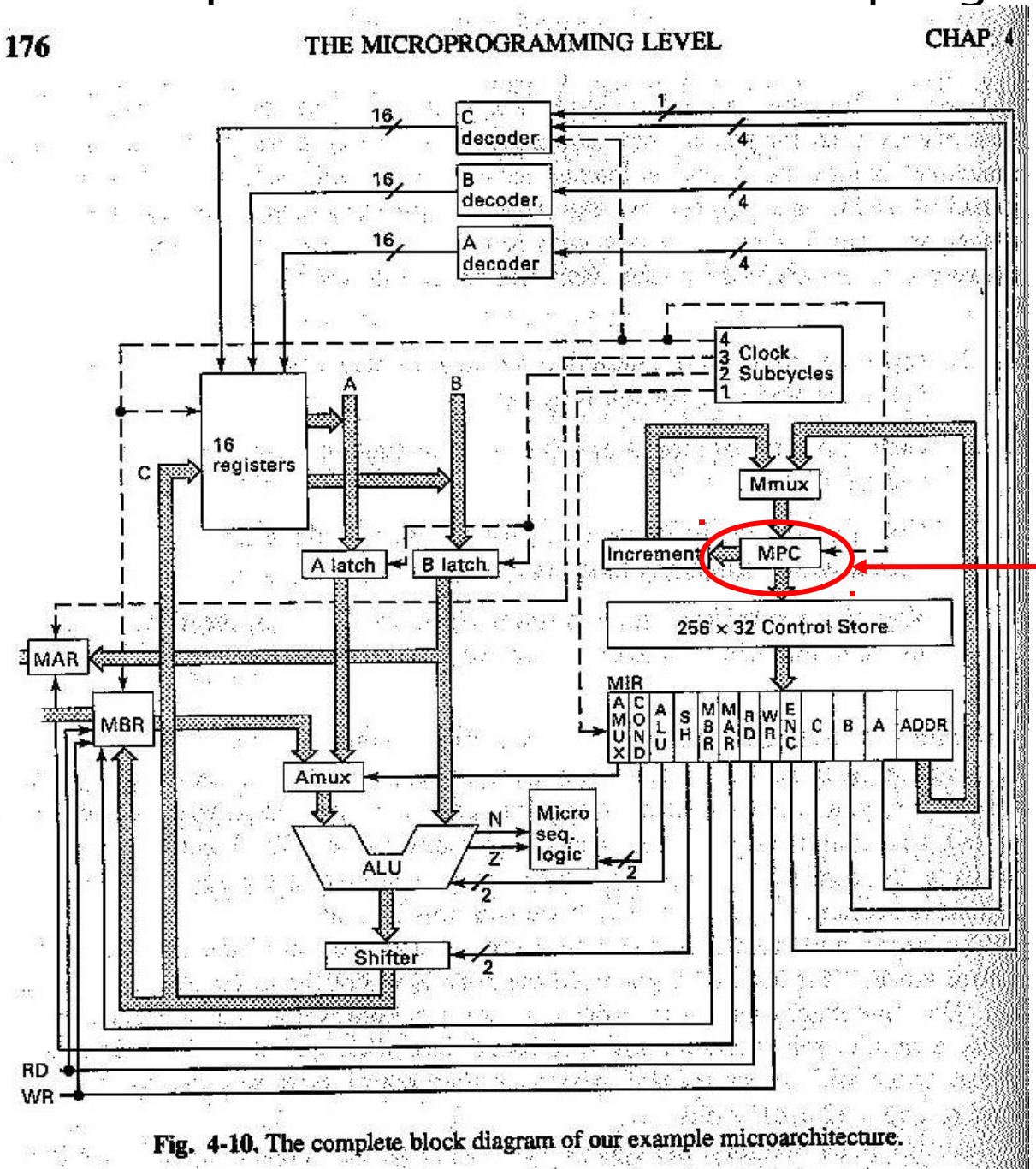


Fig. 4-10. The complete block diagram of our example microarchitecture.

Memoria di controllo

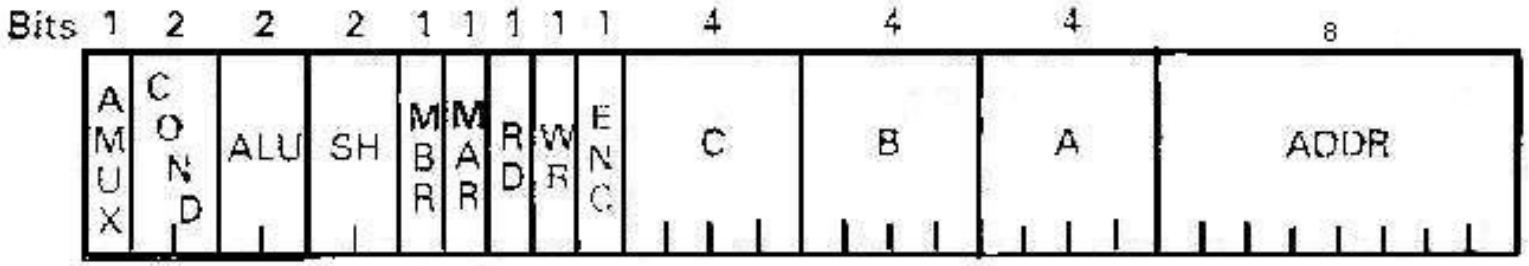
Esempio di architettura microprogrammata



Micro Program Counter

Fig. 4-10. The complete block diagram of our example microarchitecture.

Esempio di architettura microprogrammata – Formato Microistruzione



AMUX	COND	ALU	SH	MBR MAR, RD, WR, ENC
0 = A latch	0 = No jump	0 = A + B	0 = No shift	0 = No
1 = MBR	1 = Jump if N = 1	1 = A AND B	1 = Shift right 1 bit	1 = Yes
	2 = Jump if Z = 1	2 = $\overline{A}$	2 = Shift left 1 bit	
	3 = Jump always	3 = A	3 = (not used)	

Fig. 4-9. The microinstruction layout for controlling the data path of Fig. 4-8.

Esempio di architettura microprogrammata – Tempificazione

- Clock interno a 4 fasi derivato dal clock esterno
  - Fase 1 – Caricamento prossima microistruzione in MIR
  - Fase 2 – Acquisizione dei valori dei bus A e B nei registri A-latch e B-latch
  - Fase 3 – Ora che gli input all'ALU sono stabili si abilita l'ALU all'operazione
  - Fase 4 – Ora che il risultato dell'operazione è stabile in Shifter si abilita la scrittura del risultato

- Esecuzione di una microistruzione per ogni ciclo di clock esterno

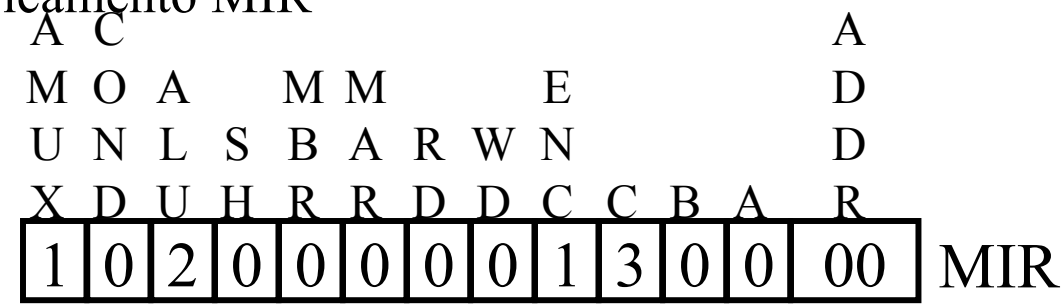
- L'esecuzione di un'istruzione macchina richiede un numero di cicli di clock pari al numero di microistruzioni necessarie

Esempio di architettura microprogrammata – Codifica microistruzioni

Microperazione	AMUX	COND	ALU	SH	MBR	MAR	RD	WD	ENC	C	B	A	ADDR
MAR:= PC; RD	0	0	2	0	0	1	1	0	0	0	0	0	00
RD	0	0	2	0	0	0	1	0	0	0	0	0	00
IR:=MBR	1	0	2	0	0	0	0	0	1	3	0	0	00

Esempio di architettura microprogrammata – Esecuzione delle microistruzioni

- Esempio IR:=MBR
  - Fase 1 – Caricamento MIR



  - Fase 2 – Caricamento dei registri A-latch e B-latch. In questo caso i valori acquisiti non verranno utilizzati per l'esecuzione della microistruzione
  - Fase 3 – Abilitazione ALU
    AMUX=1 seleziona il registro MBR
    ALU=2 lascia passare il dato senza effettuare modifiche
  - Fase 4 – Scrittura del risultato
    ENC = 1 forza la scrittura nei registri interni
    C = 3 seleziona il registro IR

Cisc (Complex Instruction Set Computer)

- Molte istruzioni potenti
- Consentono una traduzione più immediata dai linguaggi ad alto livello
- Il programma assembler è costituito da un minor numero di linee di codice
- Esecuzione delle istruzioni assembler complessa e gestita pertanto con microcodice
- Occorrono più cicli di clock per eseguire un istruzione
- È difficile realizzare compilatori ottimizzanti

Risc (Reduced Instruction Set Computer)

- Poche istruzioni e poco
- Consentono una traduzione meno immediata dai linguaggi ad alto livello
- Il programma assembler è costituito da un maggior numero di linee di codice
- Esecuzione delle istruzioni assembler smplice e gestita pertanto con logica cablata
- Occorrono pochi cicli di clock per eseguire una istruzione
- È facile realizzare compilatori ottimizzanti

Risc (Reduced Instruction Set Computer)

- Si fa forte uso della pipeline
- Rimane più spazio sul chip per realizzare registri
  - Si riducono gli accessi in memoria
  - Maggiori prestazioni
- Più operazioni di fetch …

Architetture CISC e RISC

- CISC (Complex Instruction Set Computer)
  - Utilizzo del *transistor budget* per massimizzare la taglia dell'instruction set
  - Architettura microprogrammata
  - Esempi: Intel X86, Pentium, P6

- RISC (Reduced Instruction Set Computer)
  - Utilizzo del *transistor budget* per velocizzare un repertorio limitato di istruzioni
  - Architettura non microprogrammata
  - Esempi: MIPS RX000, SPARC, IBM PowerPC