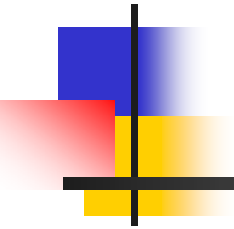
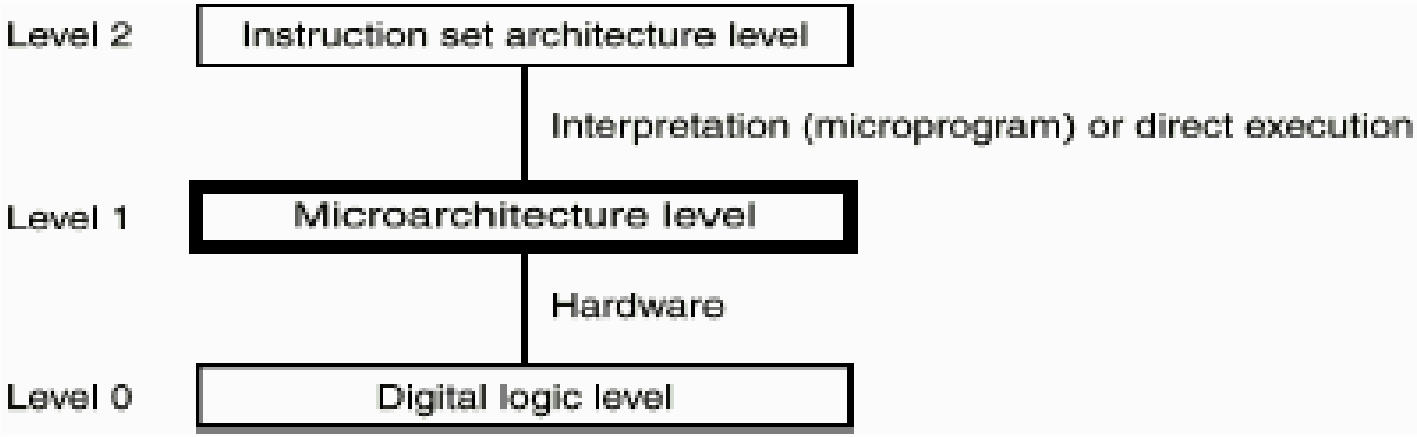


Calcolatori Elettronici (corso per Elettronici)

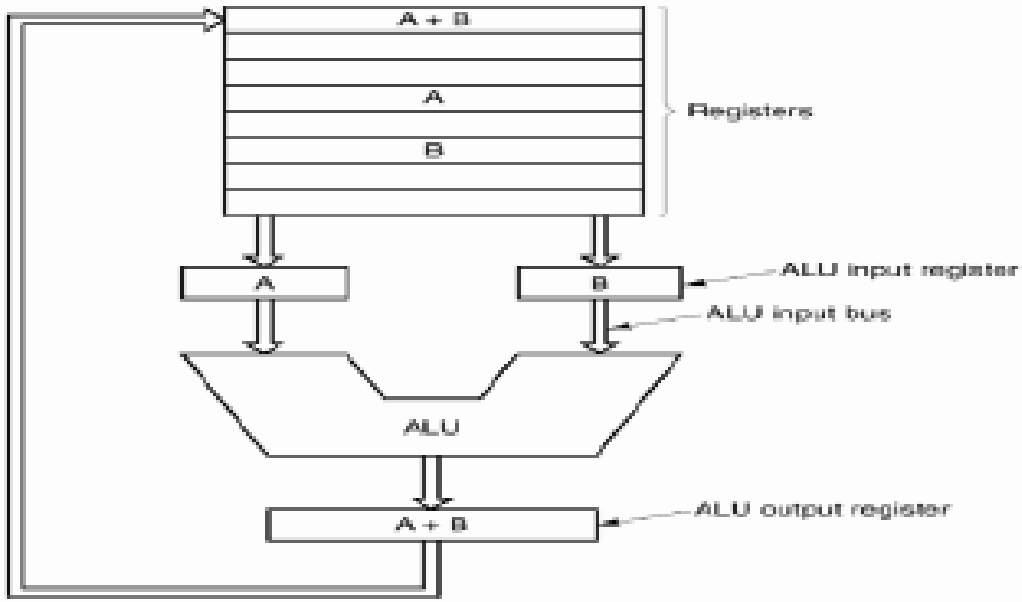
Parte VII: Microarchitettura

Il livello della microarchitettura



- Al livello della **microarchitettura** studiamo come la CPU "implementa" le istruzioni macchina mediante i dispositivi digitali (l'*hardware*) a sua disposizione
- La descrizione considera i componenti di base della CPU (registri, ALU, ecc.) e il flusso dei dati tra di essi trascurandone i dettagli realizzativi

Microarchitettura e data path



- La microarchitettura della CPU è tipicamente composta da alcuni registri, una ALU, dei bus interni e alcune linee "di controllo"
- Le istruzioni macchina comandano il funzionamento della CPU e il percorso dei dati (data path)

Possibili implementazioni

Esecuzione diretta delle istruzioni (RISC)

- Le istruzioni possono venire eseguite direttamente dalla microarchitettura
- Pro e contro:
  - Repertorio di istruzioni limitato
  - Progettazione dell'HW complessa
  - Esecuzione molto efficiente

Interpretazione delle istruzioni (CISC)

- La microarchitettura sa eseguire direttamente solo alcune semplici operazioni
- Ciascuna istruzione è scomposta in una successione di operazioni base poi eseguite dalla microarchitettura
- Pro e contro:
  - Repertorio di istruzioni esteso
  - HW più compatto
  - Flessibilità di progetto

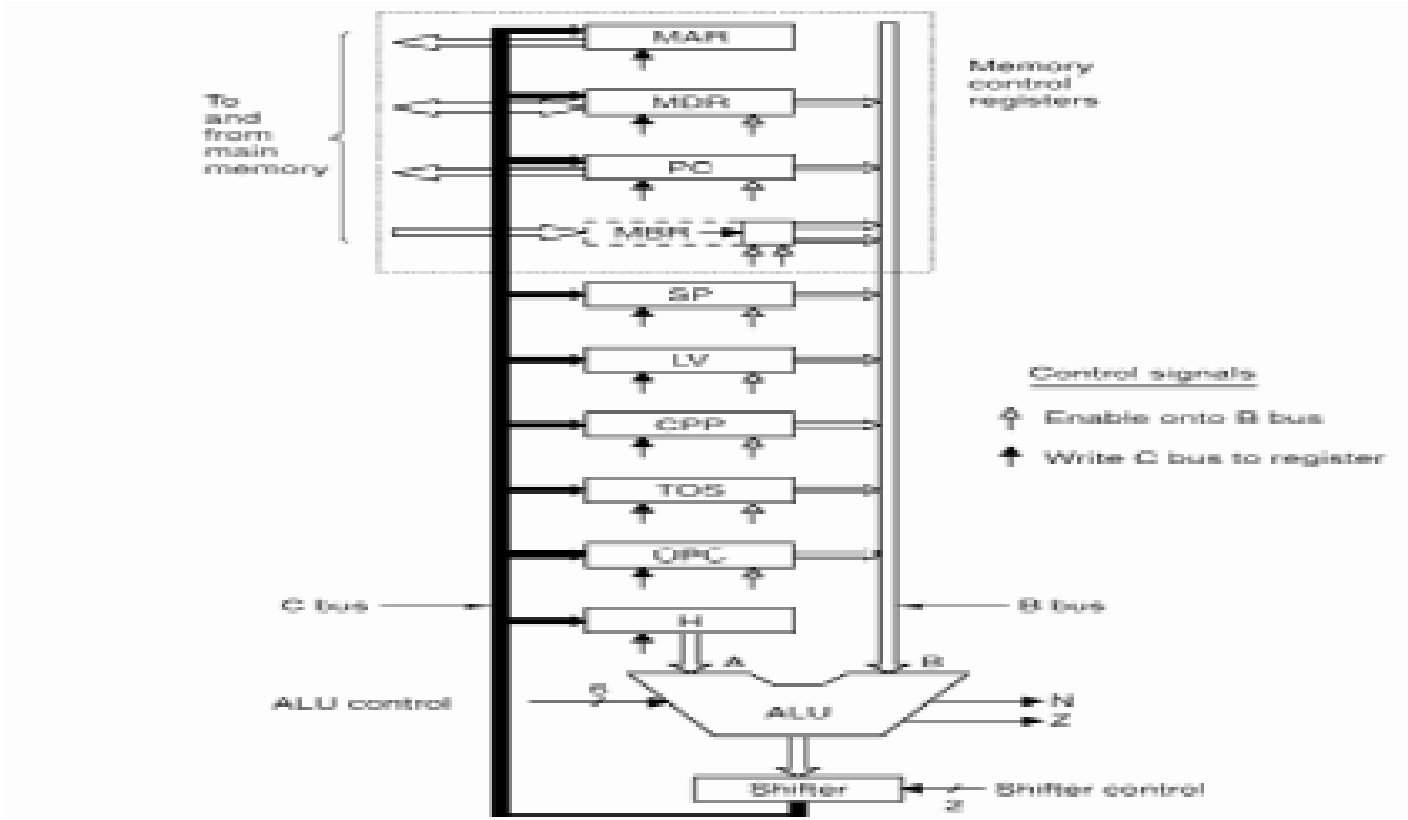
La microprogrammazione

- In un'architettura microprogrammata le istruzioni macchina non sono eseguite direttamente dall'hardware
- L'hardware esegue istruzioni a livello più basso: microistruzioni
- All'esecuzione di ciascuna istruzione macchina corrisponde l'esecuzione di diverse microistruzioni (op. elementari)
- Di fatto viene eseguito un programma, detto microprogramma, i cui dati sono le istruzioni macchina, e il cui risultato è l'interpretazione di tali istruzioni
- Vantaggi: flessibilità, possibilità di gestire istruzioni macchina complesse
- Svantaggi: esecuzione relativamente lenta; ciascuna istruzione richiede più fasi elementari

Un esempio di microarchitettura

- Implementazione di un JVM (Java Virtual Machine) con sole istruzioni su interi
- In questo corso ci limitiamo a:
  - La microarchitettura (data path)
  - La temporizzazione di esecuzione
  - L'accesso alla memoria (cache)
  - Il formato delle μ-istruzioni
  - La sezione di controllo
- Sul libro l'esempio è sviluppato fino alla definizione di un μ-programma completo per una JVM (con aritmetica intera)
- Questa ultima parte non fa parte del programma

Il cammino dei dati nella JVM

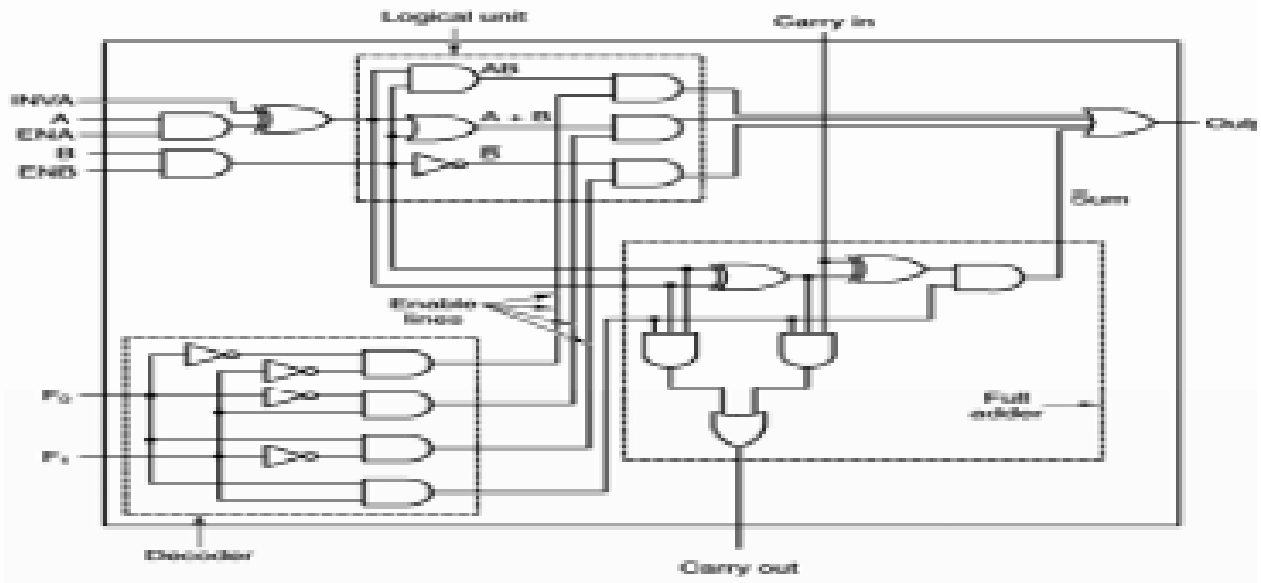


Il cammino dei dati (2)

- **Registri:** contraddistinti da nomi simbolici ciascuno con una precisa funzione
- **Bus B:** presenta il contenuto di un registro all'ingresso B della ALU
- **ALU:** ha come ingressi il bus B e il registro H (*holding register*)
- **Shifter:** consente di effettuare vari tipi di *shift* sull'uscita della ALU
- **Bus C:** permette di caricare l'uscita dello *shifter* in uno o più registri

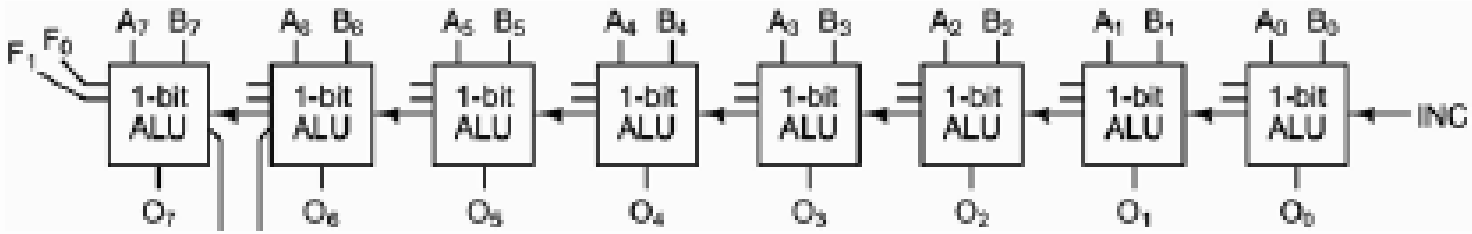
- **Segnali di controllo:**
  - *B bus enable*: trasferisce il contenuto di un registro sul bus B
  - *Write C bus*: trasferisce il contenuto dello *shifter* in uno o più registri
  - Controllo della ALU: seleziona una delle funzioni calcolabili dalla ALU
  - Controllo dello shifter: specifica se e come scalare l'uscita della ALU

Utilizziamo la ALU vista



- A e B sono bit omologhi degli operandi
- F0 e F1 selezionano la funzione (00: AND), (01: OR), (10: NOT), (11: SUM)
- ENA ed ENB sono segnali di enable e INVA permette di negare A
- Default ENA=ENB=1 e INVA=0

L’ALU è a 32 bit



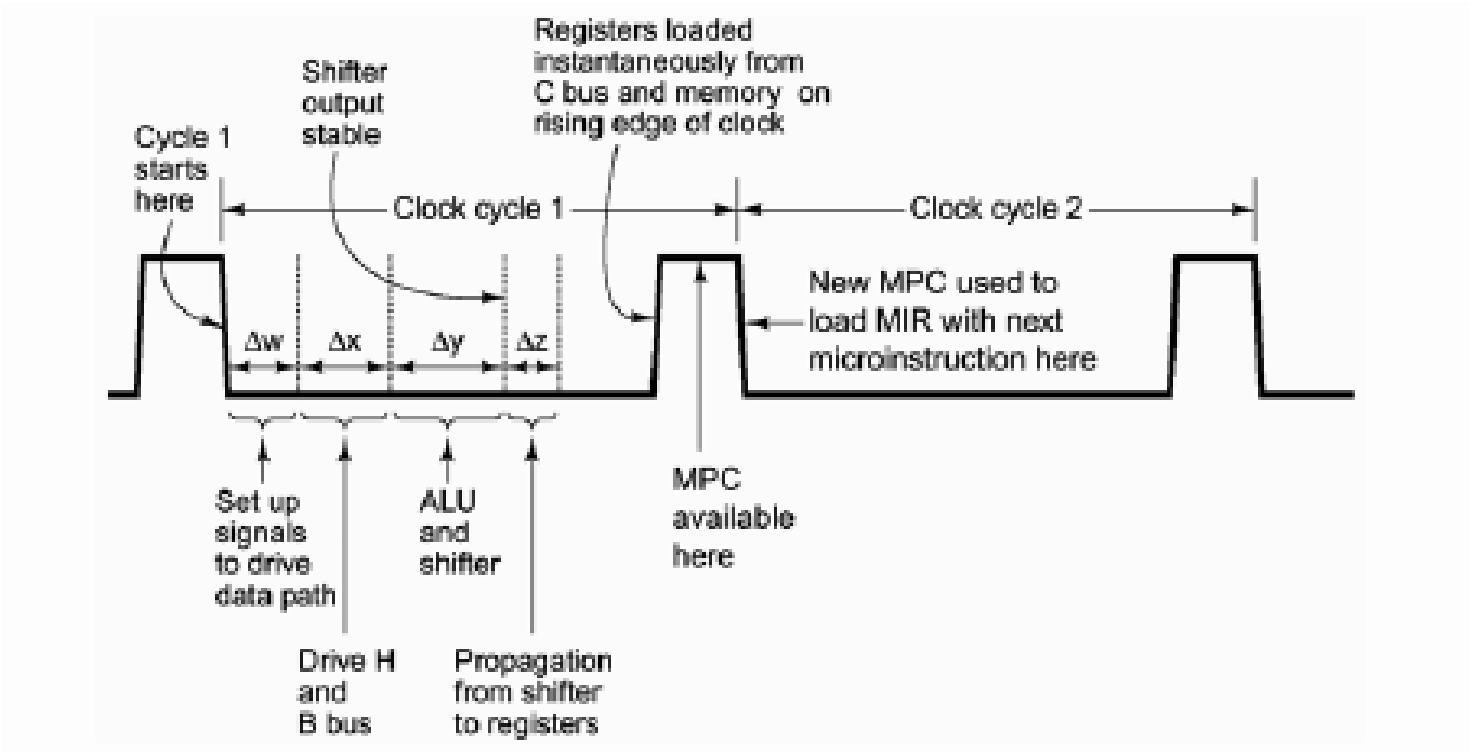
- Realizzata connettendo 32 ALU ad 1 bit (bit slices)
- INC incrementa la somma di 1 (A+1, A+B+1)

Funzioni della ALU

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	1 (0)	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
0	1	0	0	0	1	1
0	1	0	0	1	0	–1

- ENA e ENB abilitano o inibiscono gli ingressi della ALU
- INVA e INC permettono di fare il C2 di A, utile per le sottrazioni
- Possibile incrementare sia A che B e generare le costanti 0,1 e -1

Temporizzazione del ciclo base



Temporizzazione del ciclo

In ciascun ciclo di clock viene eseguita una microistruzione, cioè:

1) Caricamento di un registro sul bus B
2) Assestamento di ALU e shifter
3) Caricamento di registri dal bus C

Temporizzazione:

- Fronte di discesa: inizio del ciclo
- Δw: tempo assestamento segnali di controllo
- Δx: tempo assestamento bus B
- Δy: tempo assestamento ALU e shifter
- Δz: tempo assestamento bus C
- Fronte di salita: caricamento registri dal bus C

I tempi Δw, Δx, Δy, Δz, possono essere pensati come sottocicli (impliciti)

Accesso alla memoria

Accesso parallelo a due cache :

- Cache Dati: 32 bit indirizzabili a word (in lettura e scrittura)
- Cache Istruzioni: 8 bit indirizzabili a byte (solo in lettura)

Registri coinvolti:

- MAR (Memory Address Register): contiene l'indirizzo della word dati
- MDR (Memory Data Register): contiene la word dati
- PC (Program Counter): contiene l'indirizzo del byte di codice
- MBR (Memory Buffer Register): riceve il byte di codice (sola lettura)

Caricamento di B da parte di MBR:

- Estensione a 32 bit con tutti 0
- Estensione del bit più significativo (sign extension)

Struttura delle microistruzioni

Una μ-istruzione da 36 bit contiene:

- Tutti i segnali di controllo da inviare al data path durante il ciclo
- Le informazioni per la scelta della μ-istruzione successiva

Segnali di controllo:

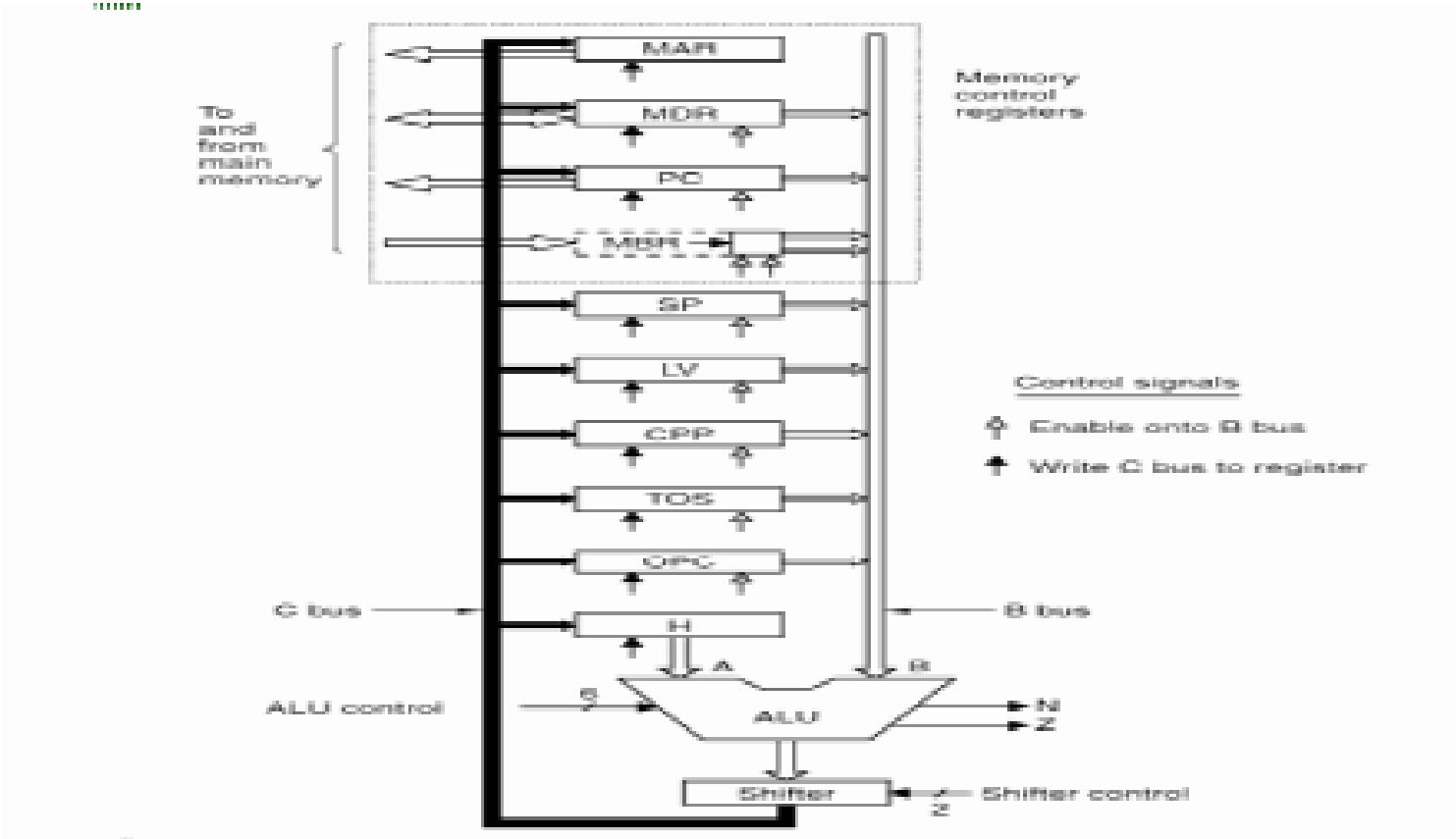
- 9 Selezione registri sul bus C
- 9 Selezione registro sul bus B
- 8 Funzioni ALU e shifter
- 2 Lettura e scrittura dati (MAR/MDR)
- 1 Lettura istruzioni (PC/MBR)

Selezione μ-istruzione successiva:

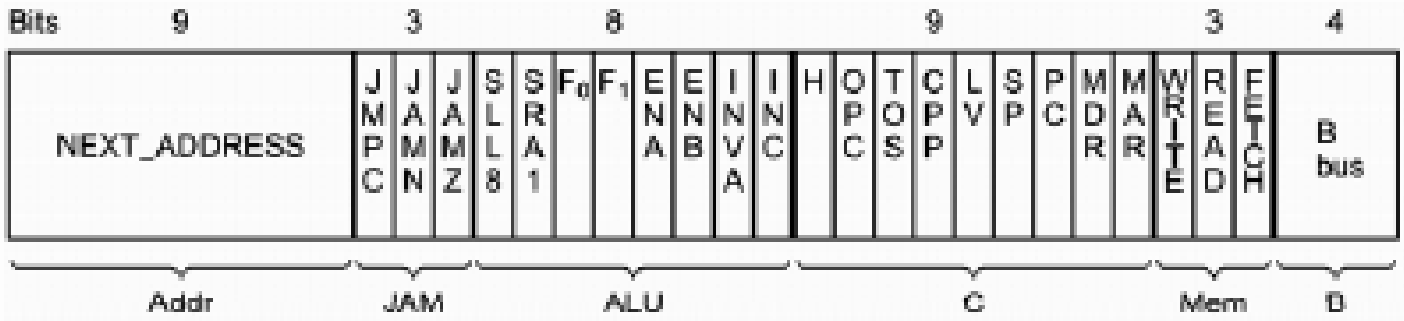
- 9 Indirizzo μ-istruzione (su 512)
- 3 Modalità di scelta

Dato che si invia su B solo un registro per volta, si codificano 9 segnali con 4

Il cammino dati nella JVM



Formato delle microistruzioni

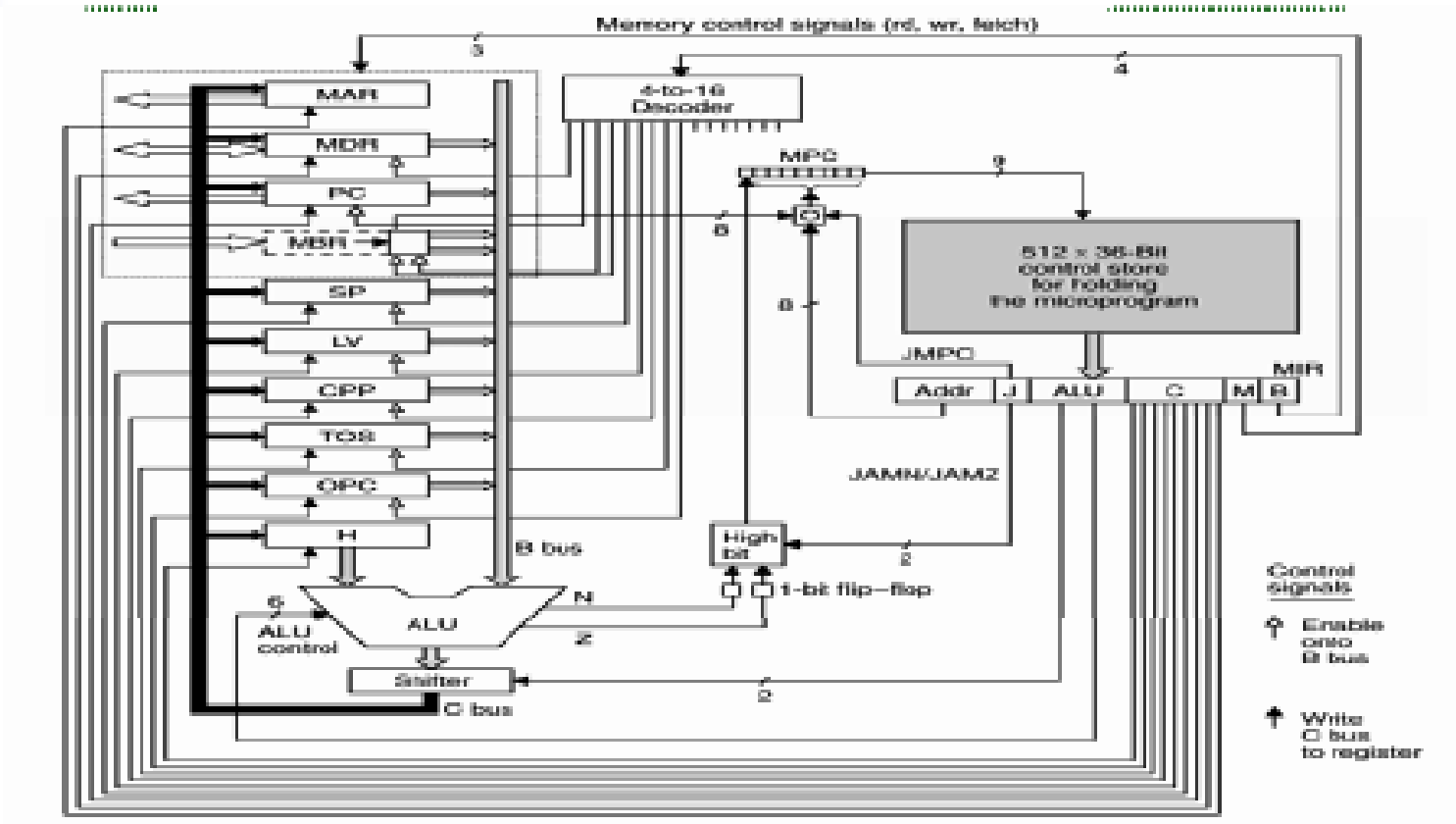


- **Addr**: Indirizzo prossima μ-istruzione
- **JAM**: Scelta prossima μ-istruzione
- **ALU**: Comandi ALU e shifter
- **C**: Registri da caricare da C
- **Mem**: Controllo memoria
- **B**: Registro da inviare su B

B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

La sezione di controllo



La sezione di controllo (2)

- Control Store: è una ROM 512×36 bit che contiene le μ-istruzioni
- MPC (Micro Program Counter): contiene l'indirizzo della prossima μ-istruzione
- MIR (MicroInstruction Register): contiene la μ-istruzione corrente
- Il contenuto di MPC diviene stabile sul livello alto del clock
- La μ-istruzione viene caricata in MIR sul fronte di discesa dell'impulso di clock
- Temporizzazione della memoria:
  - Inizio ciclo di memoria subito dopo il caricamento di MAR e di PC
  - Ciclo di memoria durante il successivo ciclo di clock
  - Dati disponibili in MDR e MBR all'inizio del ciclo ancora successivo

Temporizzazione del ciclo base