Il livello MicroArchitetturale

Corso di Architettura degli Elaboratori 1

Fulvio Valenza

A.A. 2018-2019

Polo di Vercelli

Obiettivo del livello microarchitetturale

- Obiettivo del livello microarchitetturale è l'implementazione del livello ISA (Instruction Set Architecture);
- La realizzazione "scelta" (ne esistono differenti) dipende da alcune caratteristiche fondamentali:
  - Costo,
  - Performance,
- Alcune macchine RISC hanno istruzioni talmente semplici che possono essere eseguite in un unico ciclo di clock.

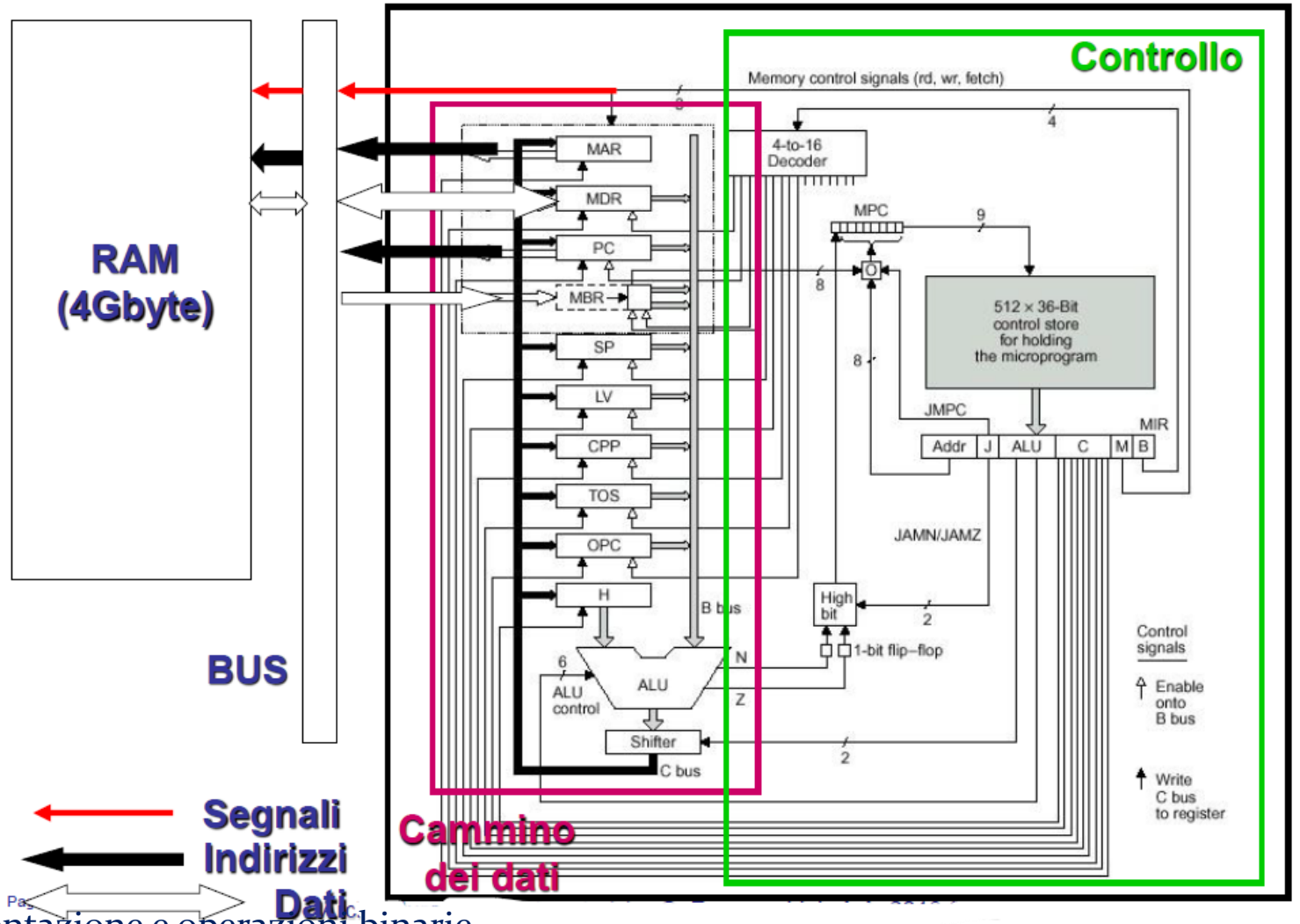
Interpretazione delle istruzioni

- 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
- Di fatto viene eseguito un programma, detto *microprogramma,* i cui dati sono le istruzioni macchina, e il cui risultato è l'interpretazione delle dette istruzioni
- **Vantaggi**: flessibilità, possibilità di gestire istruzioni macchina complesse
- **Svantaggi**: esecuzione relativamente lenta; ciascuna istruzione richiede più fasi elementari

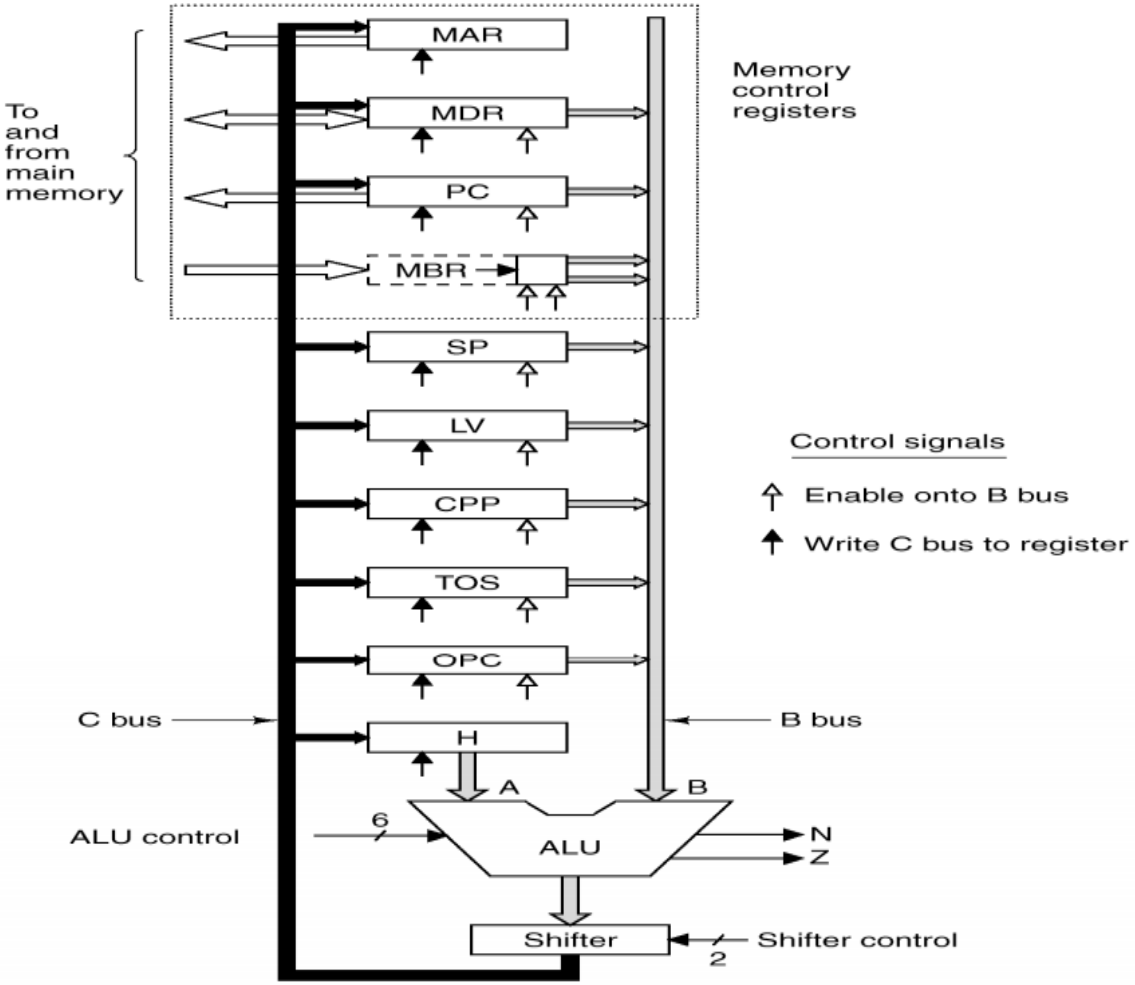
Un esempio di µ-architettura

- In questa parte del corso seguiamo un approccio operativo;
- Introdurremo i principi alla base della progettazione usando un esempio di architettura reale (un sotto insieme delle istruzione della Java Virtual Machine – IJVM)
- In questo corso ci limitiamo a:
  - La microarchitettura (data path)
  - La temporizzazione di esecuzione
  - L'accesso alla memoria (cache)
  - Il formato delle µ -istruzioni

Un esempio di μ-architettura



Il data path della microarchitettura MIC



Il datpath è ottimizzato per il particolare set di istruzioni ISA;

Ogni microistruzione controlla il datapath per un ciclo;

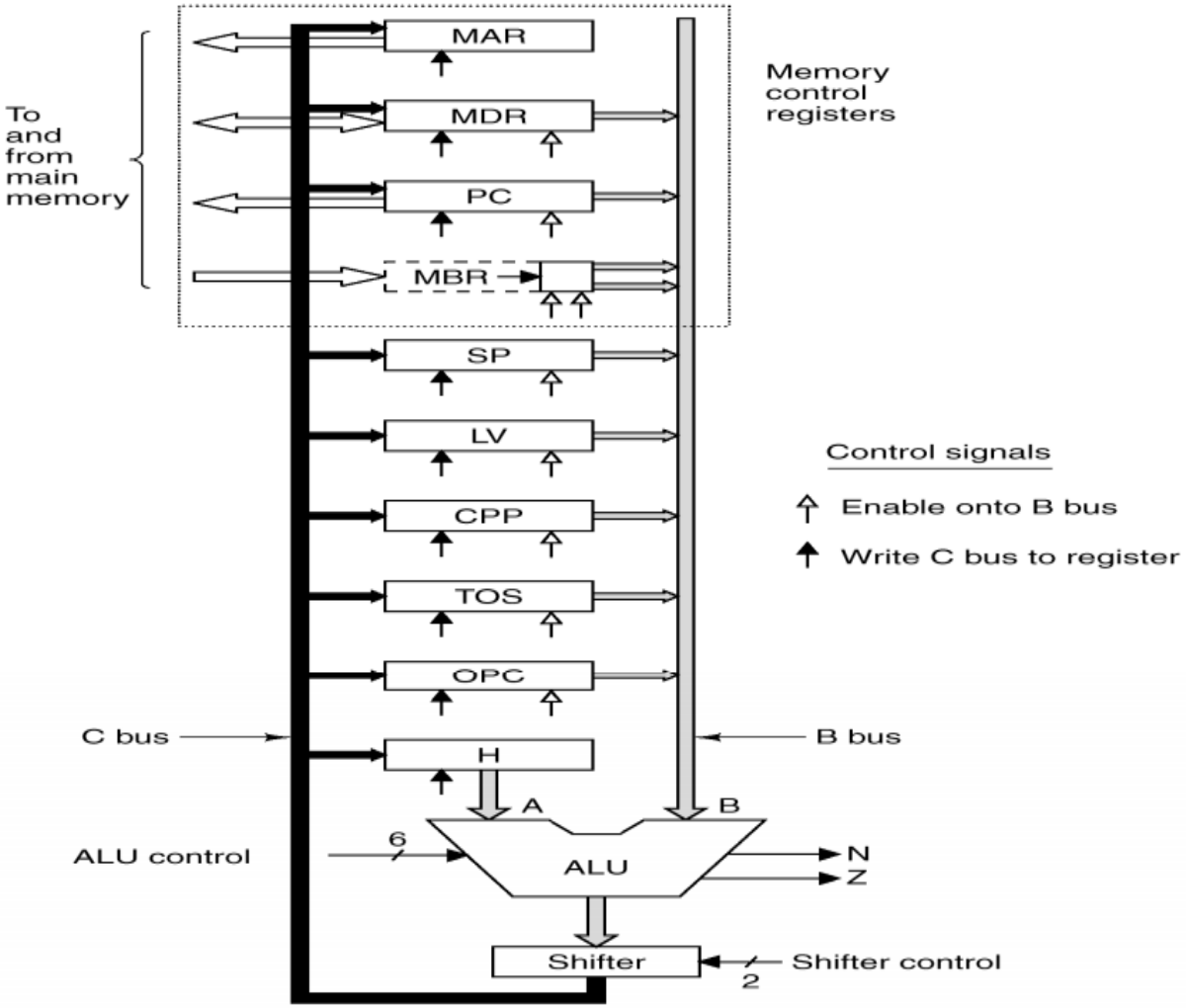
La lista di microistruzioni forma le microprocedure del microprogramma.

ALU, registri bus

La microarchitettura è costituita dalla Unità Aritmetico Logica (ALU)*, da un insieme di registri e da un insieme di bus che consentono il trasferimento dei dati*

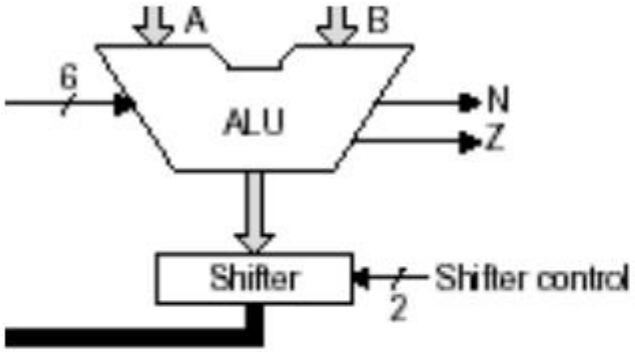
- **Registri:** contraddistinti da nomi simbolici ciascuno con una precisa funzione
- **Bus B**: presenta il contenuto di un registro all'ingresso B della ALU
- **ALU**: ha come ingressi il bus B e il registro H (*holding register*), effettua le manipolazioni dei dei dati
- **Shifter**: consente di effettuare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

ALU, registri bus



- Registri a 32 bits accessibili solo a livello microarchitetturale,
- 2 BUS (uno di lettura dai registri - B, l'altro per la scrittura sui registri - C)
- Q: come faccio a scrivere da un registro in H?

La ALU



- La ALU ha 6 linee di controllo:
- F0, F1 per l'operazione;
- ENA, ENAB per abilitare gli inputs A e B;
- INVA per invertire l'ingresso A (left - registro H);
- INC per incrementare di 1;
- Teoriche 64 combinazioni (non tutte utilizzate).
- Lo shift register ha altre due linee di controllo: Shift left logical (SLL8 - riempie il byte ms con tutti zero, Shift Right arithmetic (SRA1 - sposta di un bit e lascia invariato il bit più significativo)

La ALU

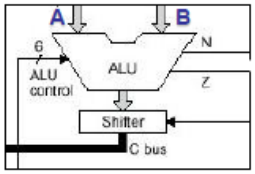
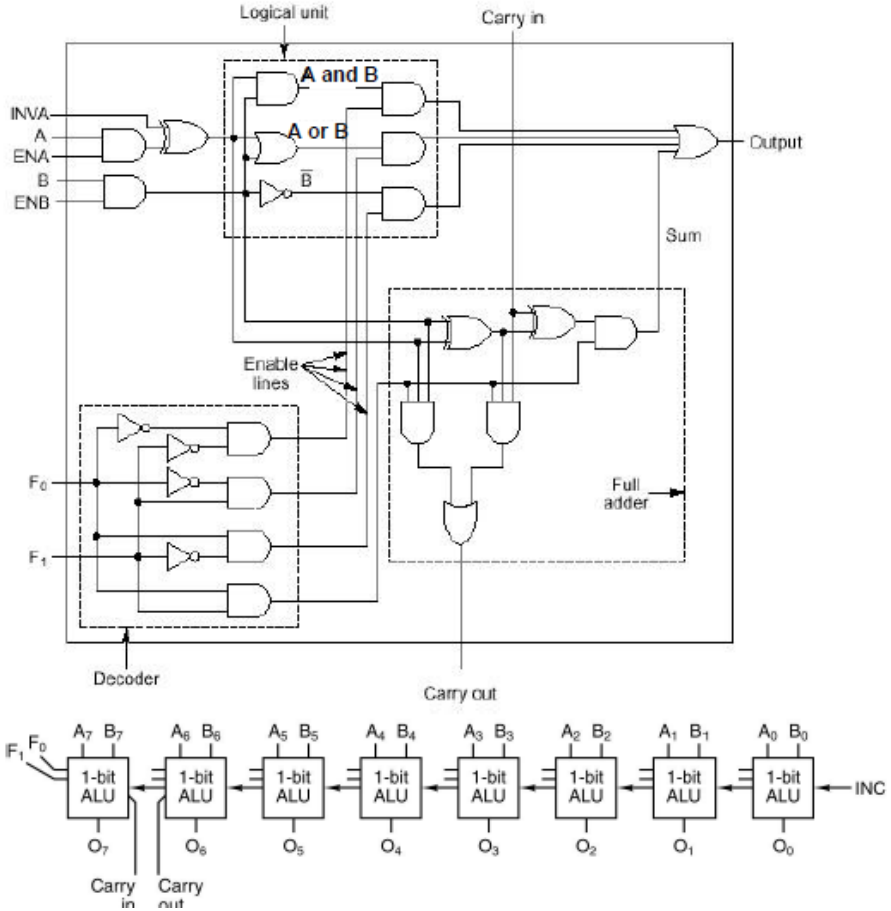


***Fig. 3.19 libro di testo*:**

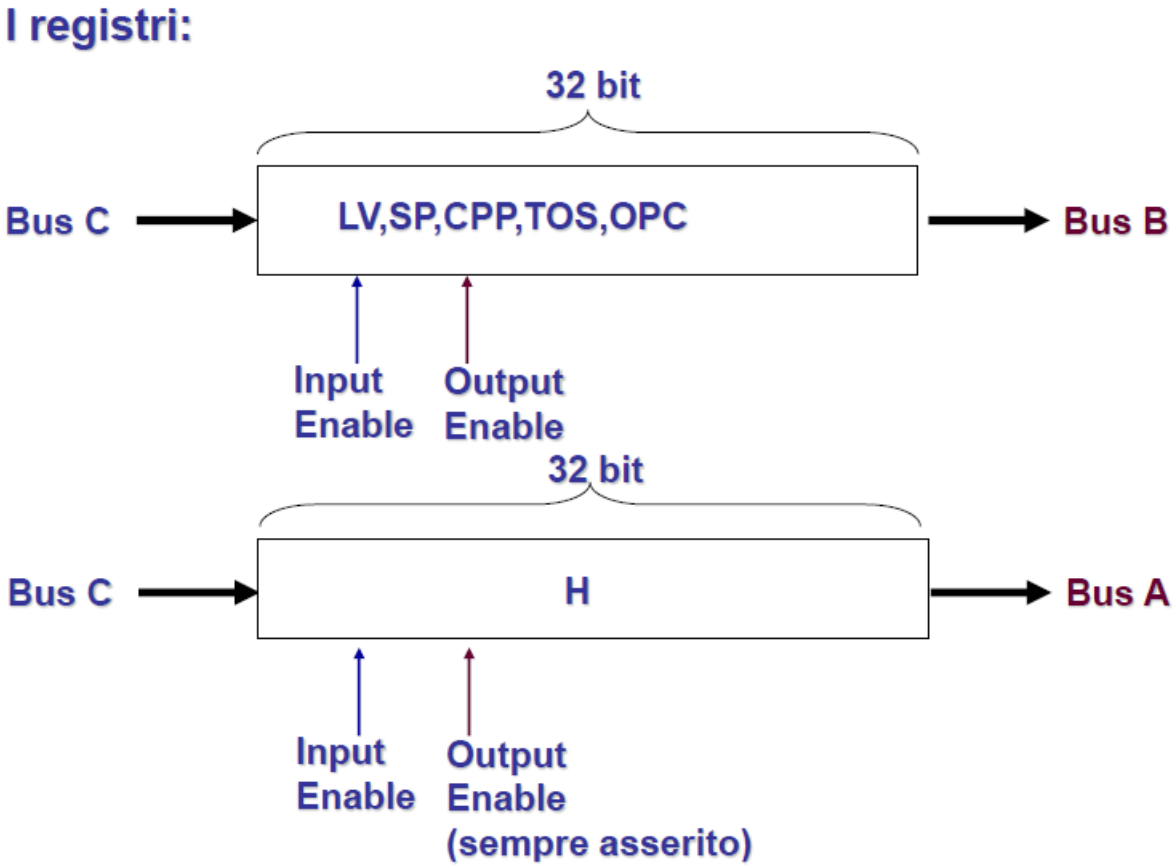
Circuito che realizza le funzioni dell'ALU (singolo bit – bit slice). INC entra come "Carry in" sul bit meno significativo. "Carry out" del bit i entra come "Carry in" del bit i+1.

***Fig. 3.20 libro di testo*:**

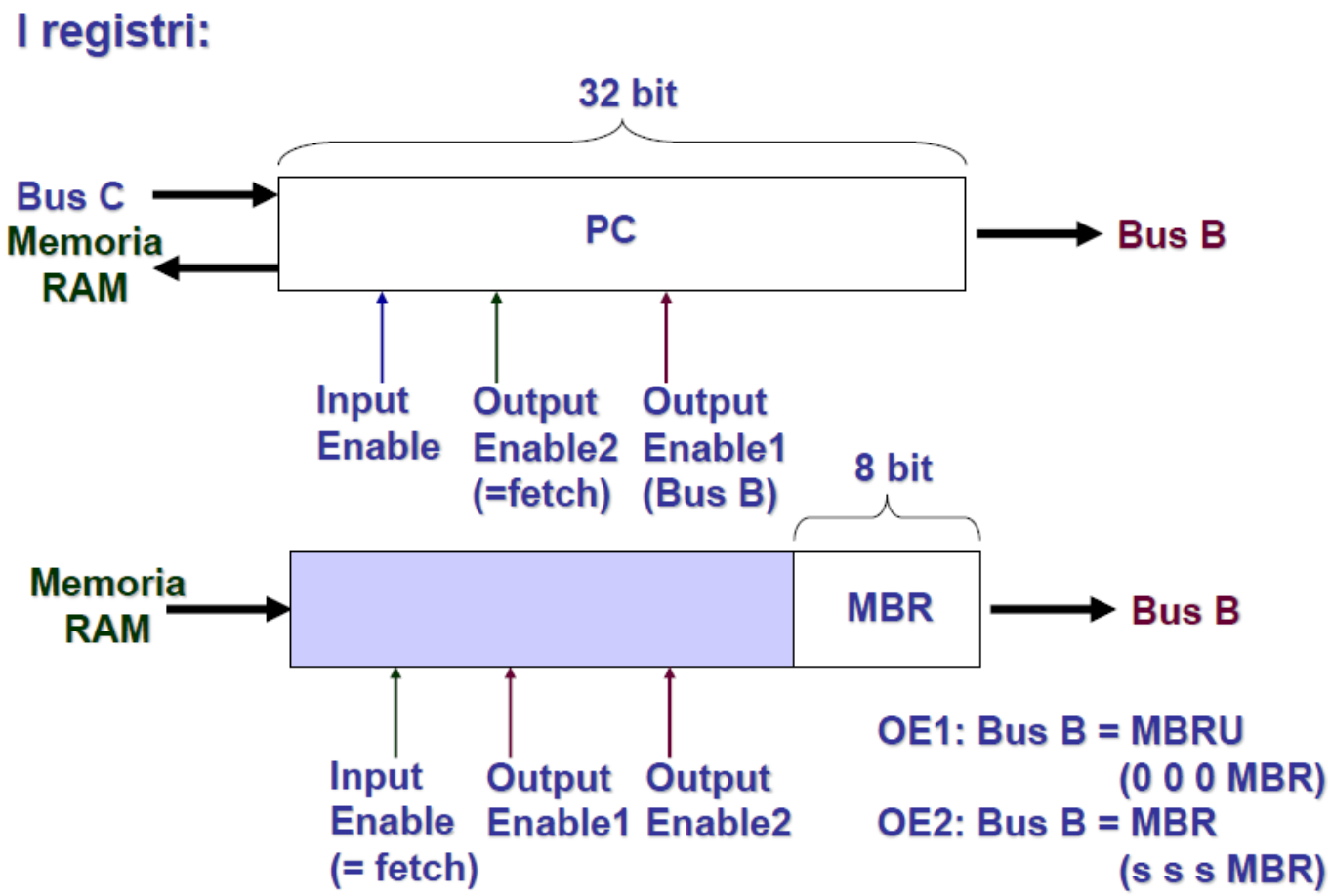
Una ALU a n bit si ottiene collegando n bit slices



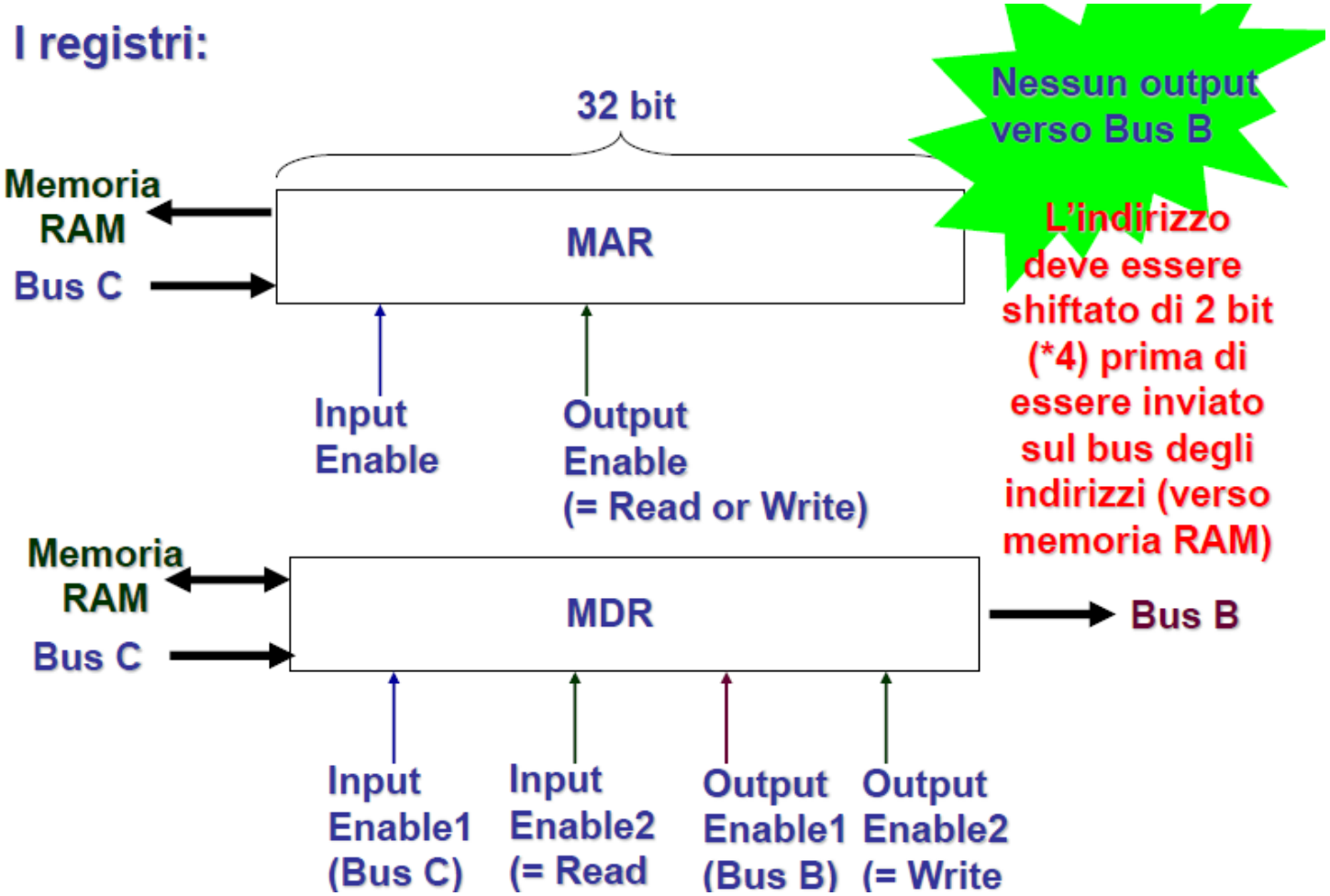
I registri


I registri:
32 bit
Bus C
LV,SP,CPP,TOS,OPC
Bus B
Input Enable
Output Enable
32 bit
Bus C
H
Bus A
Input Enable
Output Enable (sempre asserito)

I registri


I registri:
32 bit
Bus C
Memoria
RAM
PC
Bus B
Input
Enable
Output
Enable2
(=fetch)
Output
Enable1
(Bus B)
8 bit
Memoria
RAM
MBR
Bus B
Input
Enable
(= fetch)
Output
Enable1
Output
Enable2
OE1: Bus B = MBRU
(0 0 0 MBR)
OE2: Bus B = MBR
(s s s MBR)

I registri


I registri:
32 bit
Nessun output verso Bus B
Memoria RAM
Bus C
MAR
L'indirizzo deve essere shiftato di 2 bit (*4) prima di essere inviato sul bus degli indirizzi (verso memoria RAM)
Input Enable
Output Enable (= Read or Write)
Memoria RAM
Bus C
MDR
Bus B
Input Enable1 (Bus C)
Input Enable2 (= Read
Output Enable1 (Bus B)
Output Enable2 (= Write

Segnali di controllo

I trasferimenti dei dati e le funzioni svolte dalla ALU e dallo shifter sono attivate da un insieme di segnali di controllo

- **B bus enable**: trasferisce il contenuto di un dato registro sul bus B
- **Write C bus:** trasferisce il contenuto dello *shifter* in uno o più registri tramite il bus C
- **Controllo della ALU**: 6 segnali selezionano una delle funzioni calcolabili dalla ALU
- **Controllo dello shifter**: 2 segnali specificano se e come scalare l'uscita della ALU

Ulteriori connessioni permettono lo scambio di dati tra i registri MDR e MBR e due memorie cache (dati e istruzioni)

Segnali di controllo

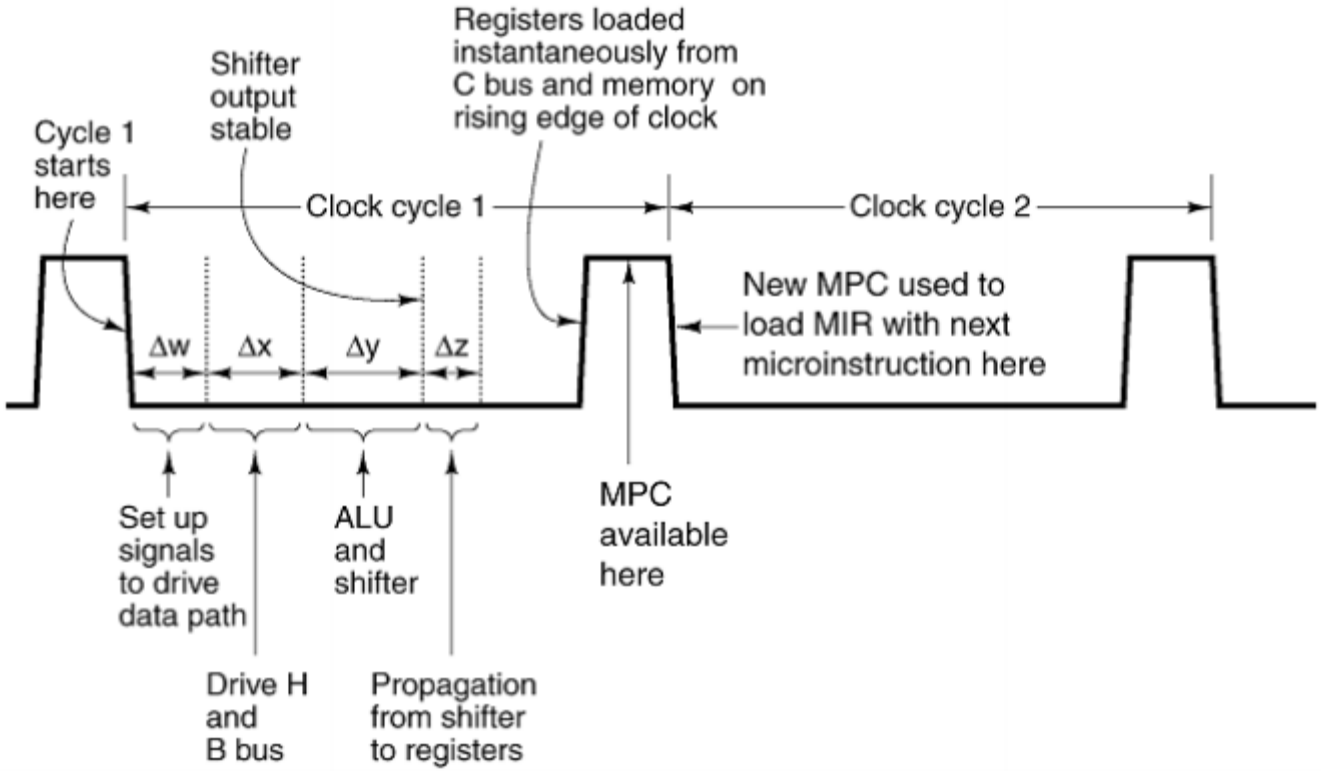
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	1	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
0	1	0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

Fa solo passare il contenuto di B

E' possibile leggere e scrivere da/e nello stesso registro in un unico ciclo di path

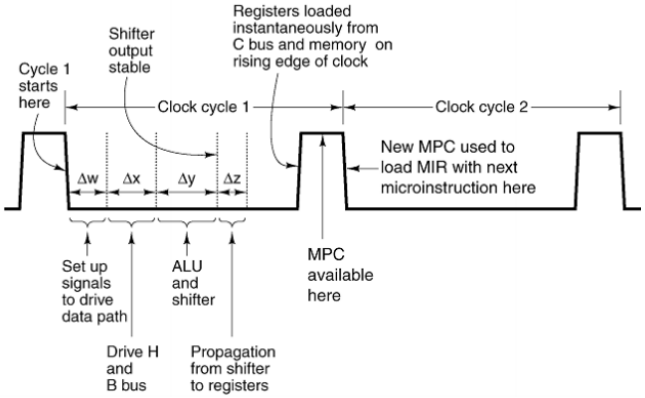
Alcune combinazioni utili dei 6 segnali della ALU e l'operazione risultante

Temporizzazione del ciclo



Temporizzazione del ciclo

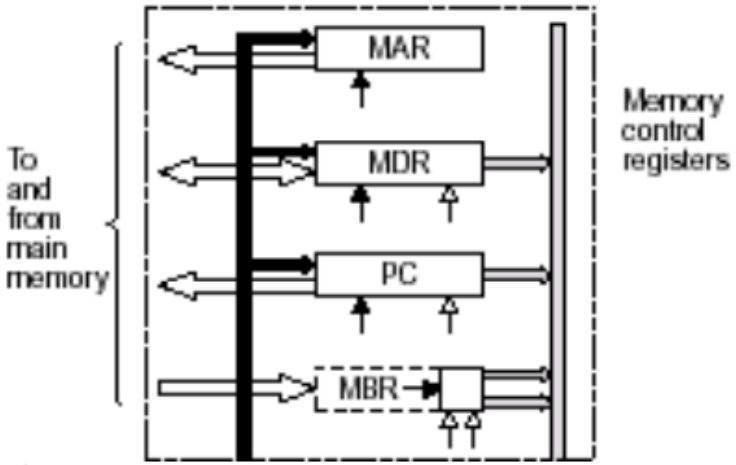
- In *ciascun ciclo di clock* viene eseguita una microistruzione:
  1) Caricamento di un registro sul bus **B**
  2) Assestamento di ALU e shifter
  3) Caricamento di registri dal bus **C**
- Temporizzazione:
  - *Fronte di discesa*: inizio del ciclo
  - **Δw**: tempo assestamento segnali di controllo
  - **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** costituiscono *sottocicli* (impliciti)



Accesso alla Memoria

- Accesso *parallelo* a due cache :
  - *Cache Dati*: 32 bit indirizzabili a *word* (lettura e scrittura)
  - *Cache Istruzioni*: 8 bit indirizzabili a *byte* (solo 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 **MBR**:
  - Estensione a 32 bit con tutti 0
  - Estensione del bit più significativo (*sign extension*)

Accesso alla Memoria



Il MIC 1 ha due modi diversi per comunicare con la memoria:

1 porto a 32 bit (indirizzabile a word) controllato da MAR/MDR;

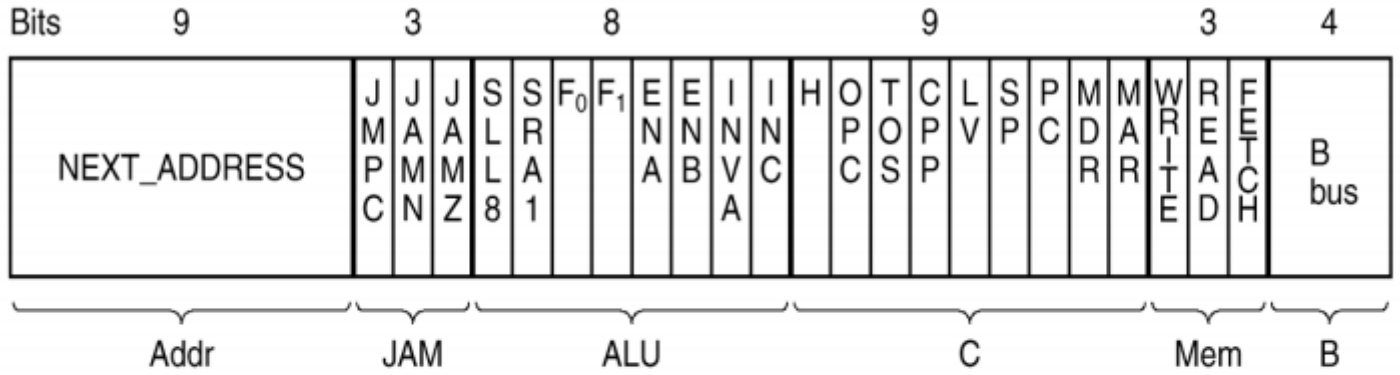
1 porto ad 8 bit (indirizzabile a byte) controllato da PC/MBR - monodirezionali;

MAR e PC indirizzano due parti differenti della memoria

Struttura delle μ-istruzioni

- Una μ-istruzione da **36 bit** contiene:
  - **A)** I segnali di controllo da inviare al *data path* durante il ciclo
  - **B)** 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*, è possibile codificare i **9** segnali di selezione registro con **4** bit

Formato delle μ-istruzioni



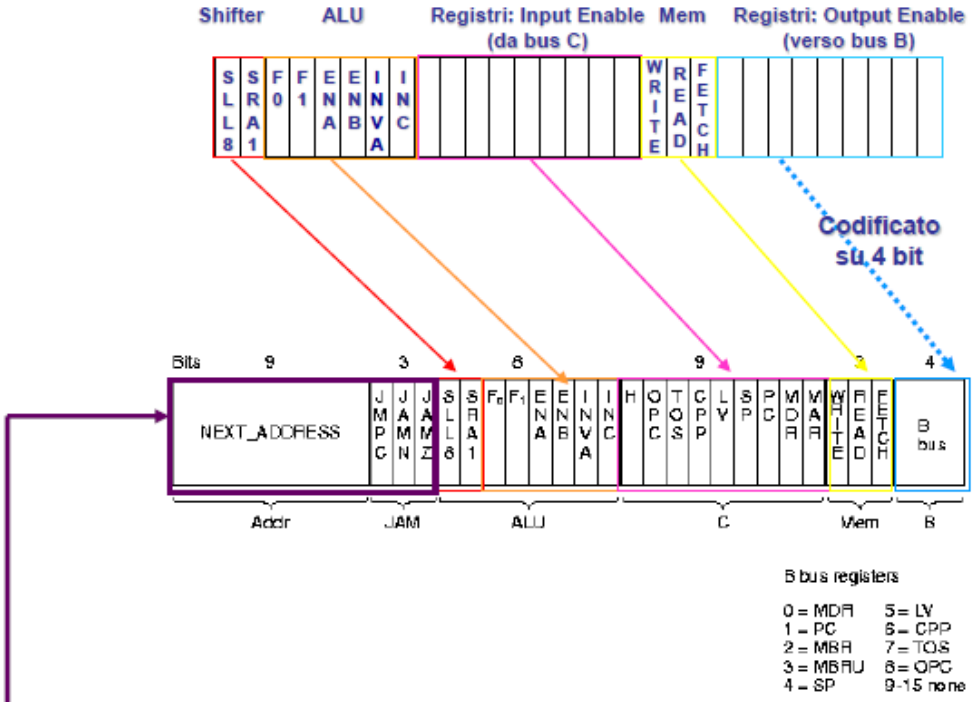
Addr - Indirizzo prossima μ-istruzione
JAM - Scelta prossima μ-istruzione
ALU - Comandi ALU e shifter
C - Registri da caricare da C
Mem - Controllo memoria
B - Registro da inviare su B

B bus registers

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

Formato delle μ-istruzioni

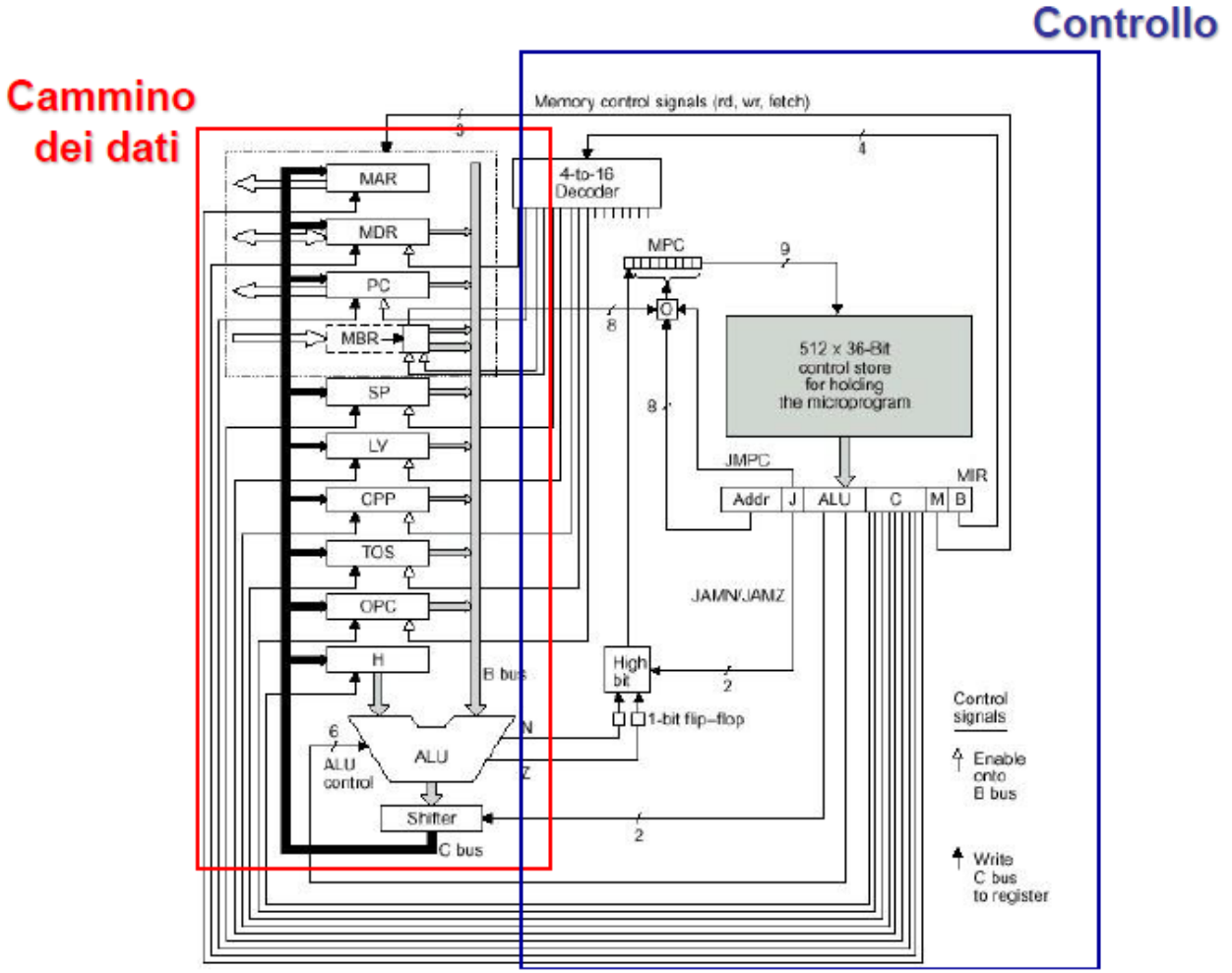
L'insieme di segnali che controllano il data path costituiscono una istruzione per la corrispondente macchina (hardware)



Le 9 configurazioni possibili dell'ultimo campo possono essere codificate in un numero inferiore di bit ... al prezzo di dover introdurre un decodificatore per generare i 9 segnali OE da mandare ai registri.

Per poter eseguire un programma in questo linguaggio occorre aggiungere un meccanismo per il controllo del flusso

L’interazione tra Cammino dei dati e Controllo



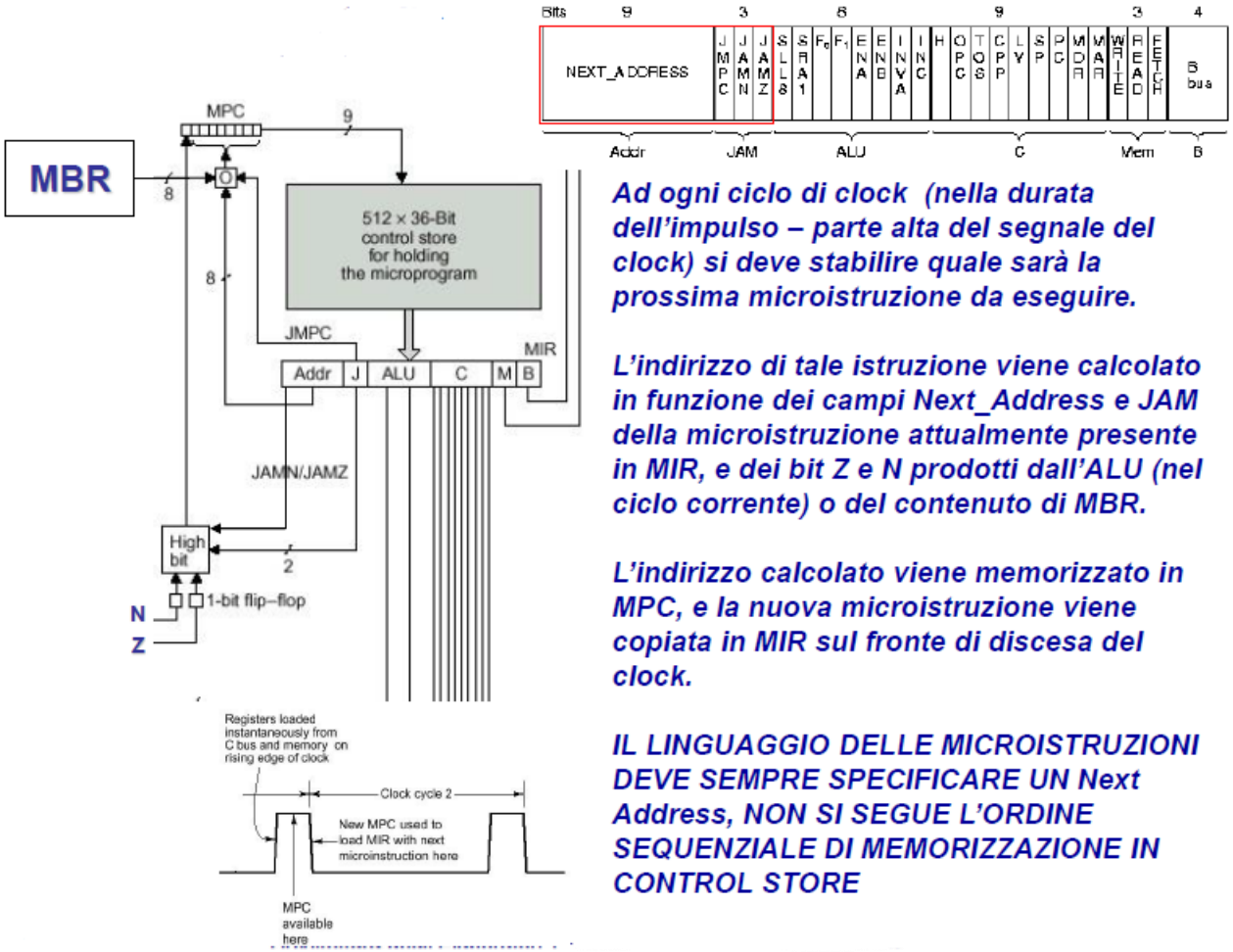
La Sezione di Controllo

- La parte principale è costituita da una *control store* di 512 word da 36 bits.
- Le microistruzioni non sono memorizzate in ordine, serve un meccanismo più flessibile (dato che le sequenze delle microprocedure sono brevi, si può implementare un meccanismo più efficiente): ogni microistruzione esplicita il suo successore.

La Sezione di Controllo

- Per accedere alla control store si usano due registri MPC (anche se non serve un contatore) e MIR;
- Il MIR viene caricato sul fronte di discesa del clock, poi le fasi del clock procedono come visto nel diagramma delle tempificazioni.
- Quando la ALU ha completato la sua elaborazione, lo **shift register** ed **N** e **Z** sono stabili; N e Z vengono salvati in una coppia di 1-bit flip flops sul fronte di salita del clock.
- N e Z vengono usati per calcolare la microistruzione successiva

La Sezione di Controllo



Ad ogni ciclo di clock (nella durata dell'impulso – parte alta del segnale del clock) si deve stabilire quale sarà la prossima microistruzione da eseguire.

L'indirizzo di tale istruzione viene calcolato in funzione dei campi Next_Address e JAM della microistruzione attualmente presente in MIR, e dei bit Z e N prodotti dall'ALU (nel ciclo corrente) o del contenuto di MBR.

L'indirizzo calcolato viene memorizzato in MPC, e la nuova microistruzione viene copiata in MIR sul fronte di discesa del clock.

IL LINGUAGGIO DELLE MICROISTRUZIONI DEVE SEMPRE SPECIFICARE UN Next Address, NON SI SEGUE L'ORDINE SEQUENZIALE DI MEMORIZZAZIONE IN CONTROL STORE

La Sezione di Controllo (riassunto)

- **Control Store**: 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 dopo il caricamento di **MAR** e di **PC**
- Ciclo di memoria durante il successivo ciclo di μ-istruzione
- Dati disponibili in **MDR** e **MBR** all'inizio del ciclo di μ-istruzione ancora successivo

La Sezione di Controllo