Capitolo 11: Organizzazione interna dei calcolatori

Reti Logiche
Contemporary Logic Design

Randy H. Katz
University of California, Berkeley

Trasparenze tradotte da:
Luciano Lavagno

Revisione: Tiziano Villa

Motivazione

- Progetto di un calcolatore come applicazione del progetto logico
- Calcolatore = unita' centrale (CPU) + sistema di memoria
- Unita' centrale = unita' di controllo (UC) + unita' operativa (UO)
- Unita' di controllo = Macchina a Stati Finiti

  Ingressi = istruzioni da eseguire, condizioni dall'UO

  Uscite = segnali di controllo di trasferimento tra registri

  Interpretazione istruzione = prelievo ("fetch"), decodifica, esecuzione

- Unita' operativa = unita' funzionali + registri

  Unita' funzionali = ALU, moltiplicatori, divisori, ecc.

  Registri = program counter, registri a scorrimento e di memorizzazione

Riassunto del capitolo

Progetto di una unita' operativa e di una unita' di controllo

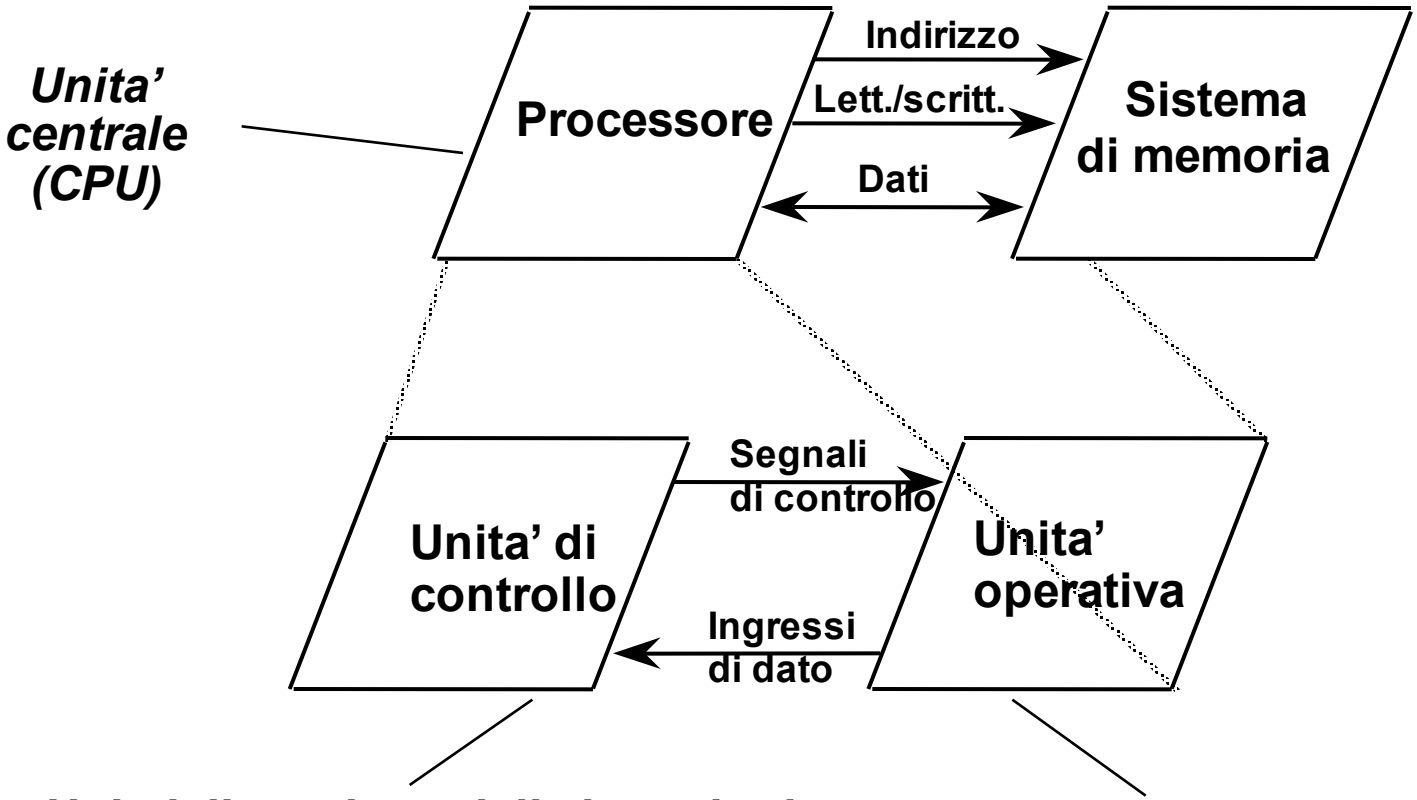
- **Metodi di interconnessione per l'UO:**

  diretta, bus singolo, bus multipli

- **Struttura del diagramma degli stati/ASM della FSM di controllo**

Struttura di un calcolatore

Schema a blocchi



Unita' di gestione delle istruzioni

FSM di fetch ed interpretazione delle istruzioni

Unita' di esecuzione

Unita' funzionali e registri

Struttura di un calcolatore

Esempio di sequenza di esecuzione delle istruzioni

Istruzione: somma Rx ad Ry e metti il risultato in Rz

Passo 1: *preleva l'istruzione di somma dalla memoria e mettila nel Registro Istruzioni (IR)*

Passo 2: *decodifica l'istruzione*

L'istruzione nell'IR e' una ADD

Gli operandi in lettura sono Rx, Ry

L'operando in scrittura e' Rz

Passo 3: *esegui l'istruzione*

Trasferisci Rx, Ry nell'ALU

Configura l'ALU per eseguire la somma

Somma Rx ad Ry

Trasferisci il risultato dell'ALU in Rz

Struttura di un calcolatore

Tipi di istruzioni

- **Manipolazione dati**

  Somma, sottrazione, ecc.

- **Preparazione e trasferimento dati**

  Lettura/scrittura (load/store) dati da/in memoria

  Trasferimenti tra registri

- **Controllo**

  Salti condizionati e non condizionati

  Chiamata a procedura e ritorno da procedura

Struttura di un calcolatore

Unita' di Controllo

Elementi dell'Unita' di Controllo (o unita' di gestione delle istruzioni):

Soliti aspetti di una FSM:

- **Registro di stato**
- **Logica di stato futuro**
- **Logica di uscita (segnali di controllo dell'UO)**

Piu' registri "di controllo" aggiuntivi:

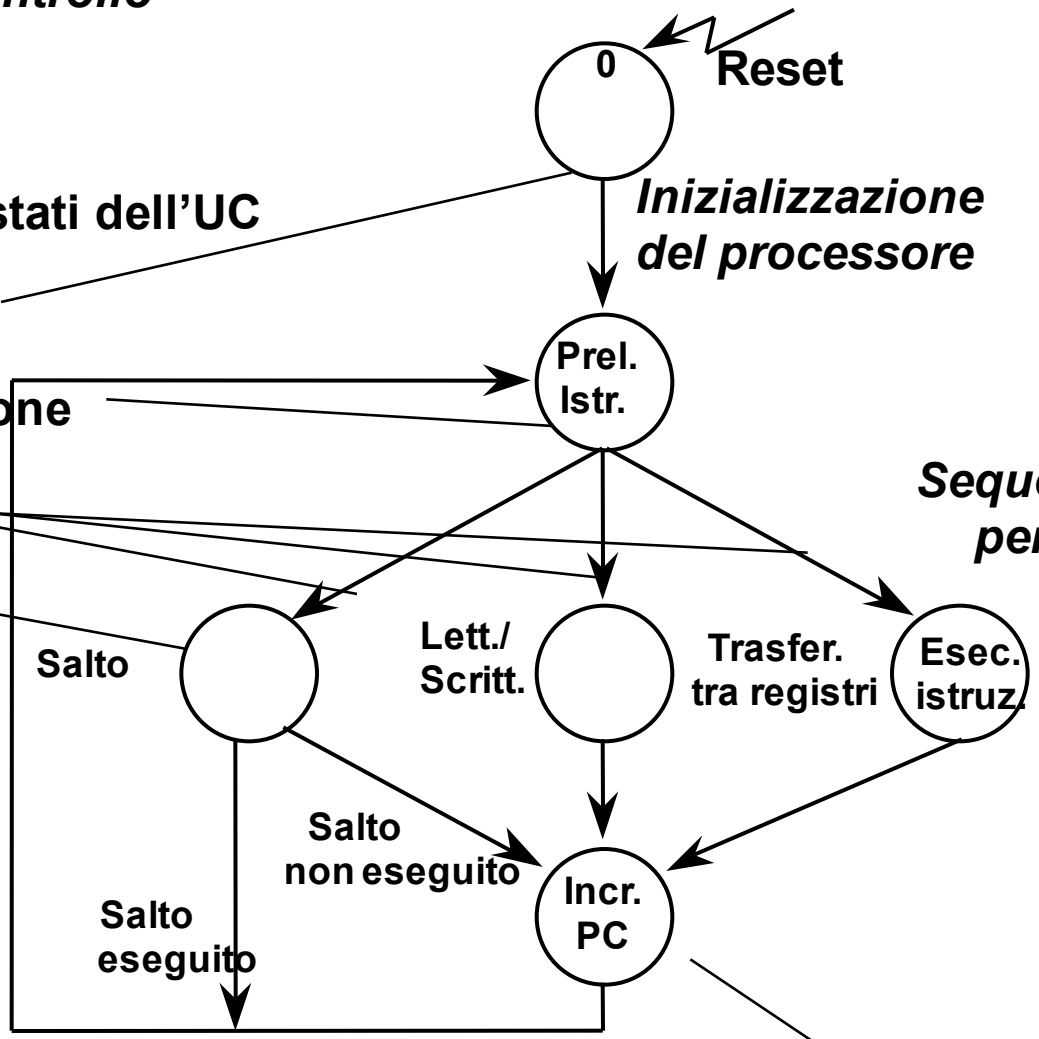
- **Registro Istruzioni (Instruction Register, IR)**
- **Contatore di programma (Program Counter, PC)**

Struttura di un calcolatore

Unita' di Controllo

Diagramma degli stati dell'UC

- **Inizializzazione**
- **Prelievo istruzione**
- **Decodifica**
- **Esecuzione**



Sequenza diversa per ogni tipo di istruzione

Istruzioni divise in tre classi:

- **Salto**
- **Lettura/scrittura**
- **Trasferimento tra registri**

Operazioni ausiliarie

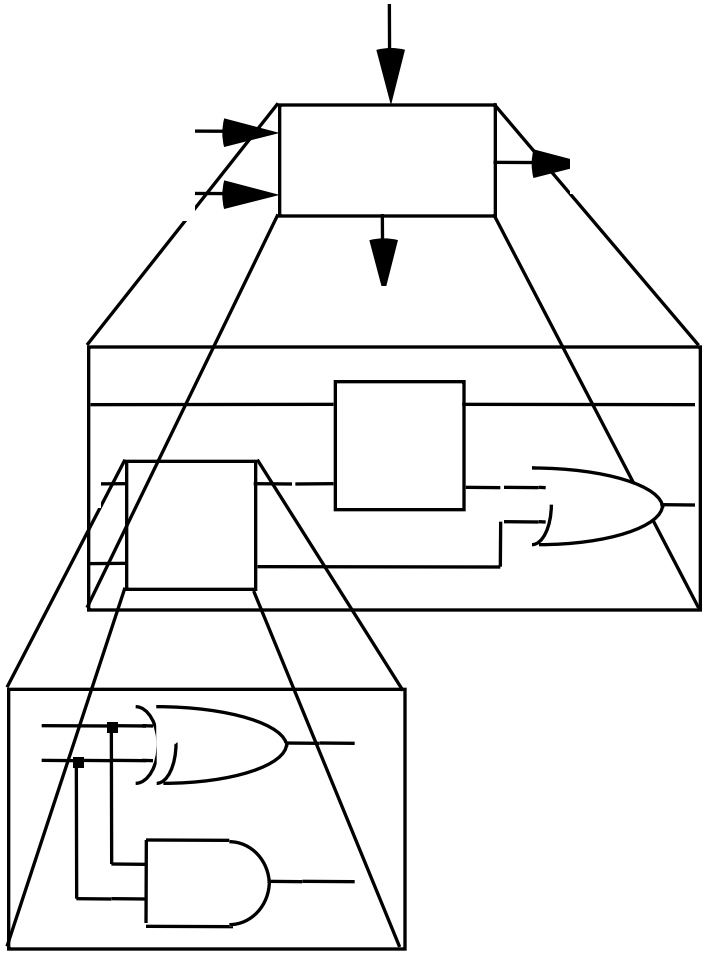
Struttura di un calcolatore

Unita' Operativa

I circuiti aritmetici sono costruiti in modo
- **gerarchico**
- **iterativo**

Tutti i bit dell'UO sono funzionalmente identici l'uno all'altro

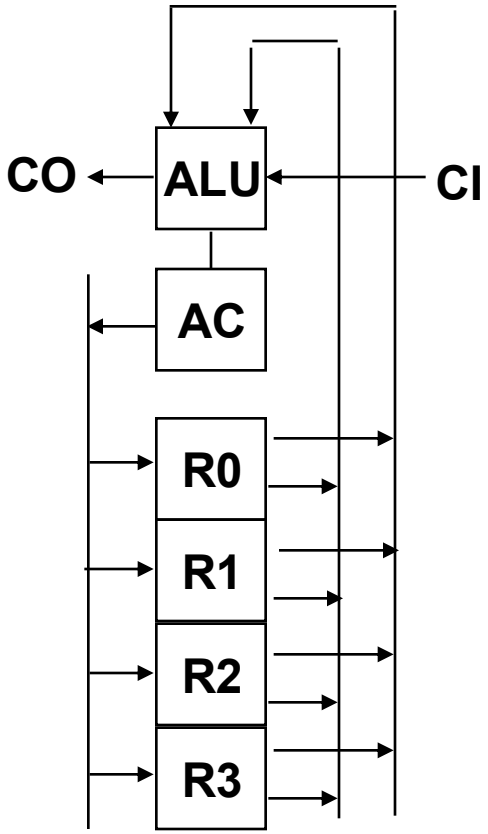
Unita' operative a:
- **4 bit**
- **8 bit**
- **16 bit**
- **32 bit**
- **64 bit**
- **(128 bit??)**



Costruzione gerarchica di un sommatore

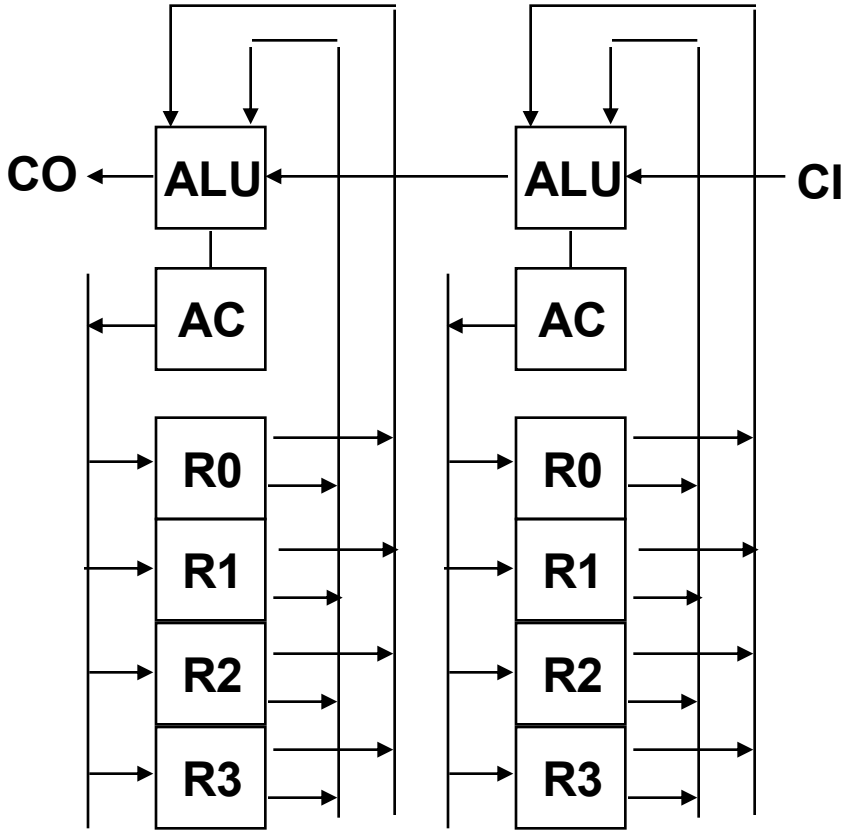
Struttura di un calcolatore

Unita' operativa



UO ad 1 bit

Concetto di "bit slice"



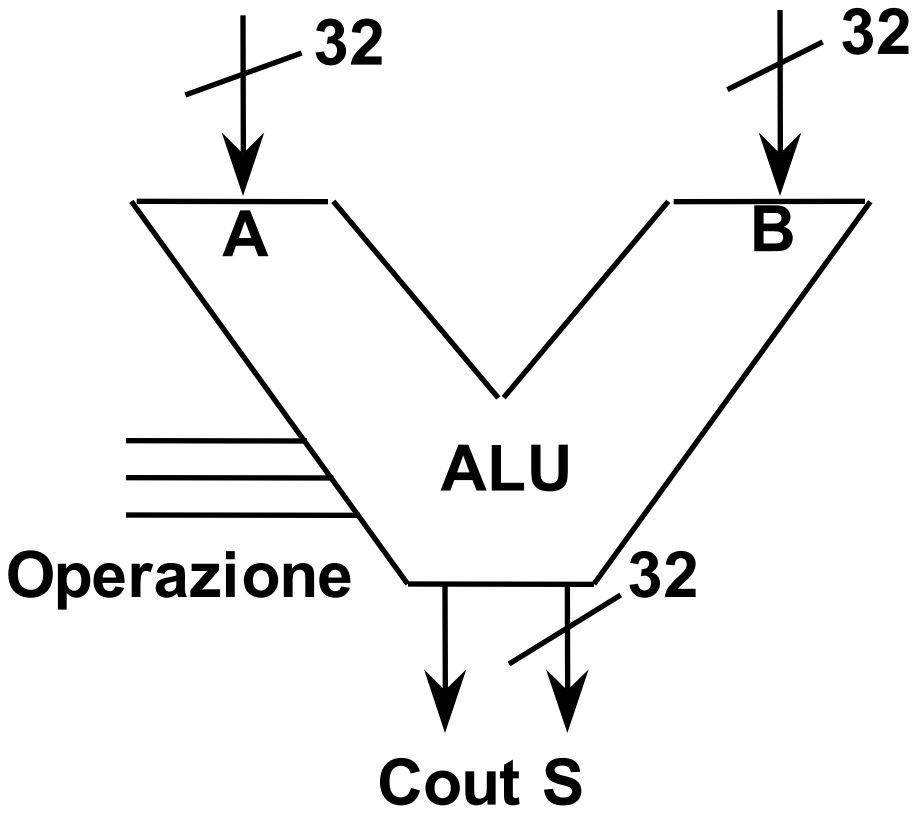
UO a 2 bit

Ripetere *n* "bit slice" per ottenere una UO ad *n* bit

Struttura di un calcolatore

Unita' operativa

Schema a blocchi dell'ALU



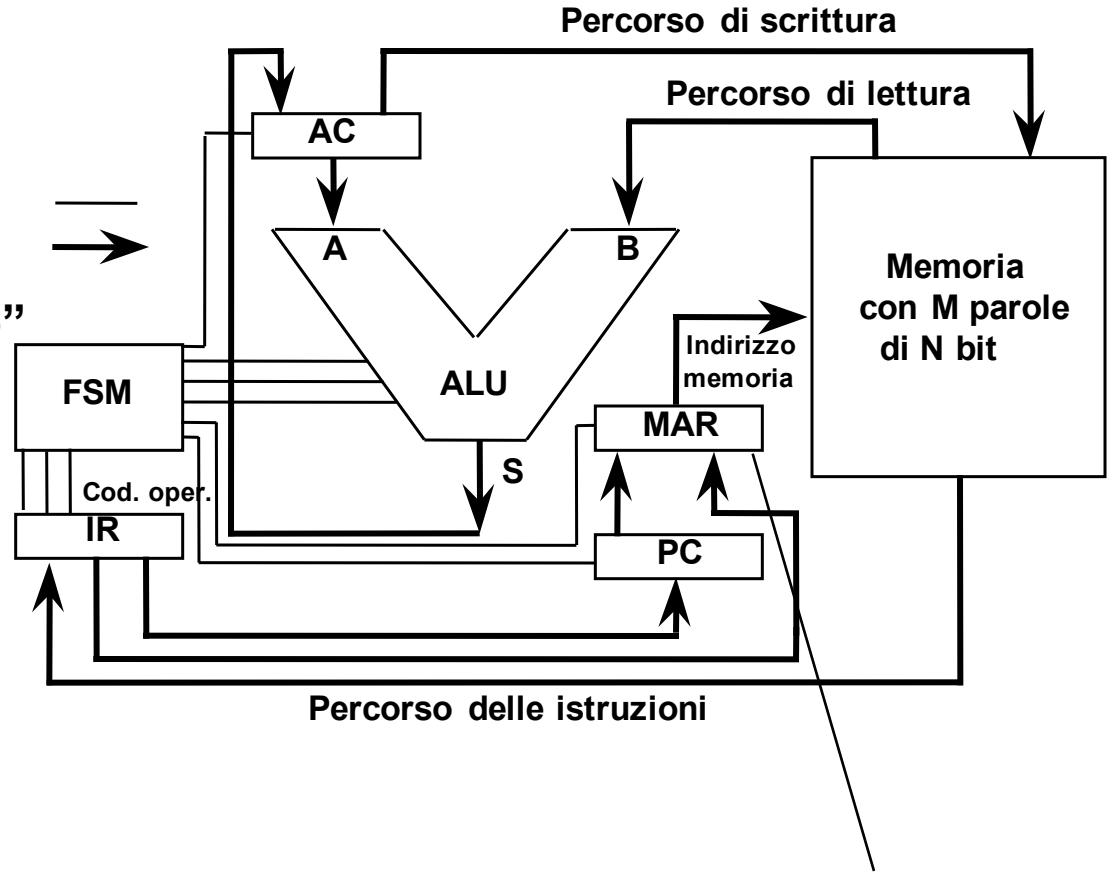
Struttura di un calcolatore

Schema a blocchi: vista a livello di trasferimento tra registri (RTL)

Macchina ad accumulatore singolo

AC := AC <op> Mem

Istruzioni "ad un solo indirizzo"
AC e' un operando implicito



Le linee con la freccia rappresentano trasferimenti di dati

Le altre sono segnali di controllo

Registro indirizzamento memoria (Memory Address Register, MAR)

Mantiene l'indirizzo stabile durante gli accessi a memoria

Struttura di un calcolatore

Schema a blocchi: vista a livello RTL

Collocamento di istruzioni e dati in memoria:

- **Dati ed istruzioni insieme in memoria: architettura di Princeton (Von Neumann)**
- **Dati ed istruzioni in memorie separate: architettura di Harvard**

L'architettura di Princeton e' piu' facile da realizzare

L'architettura di Harvard ha prestazioni migliori:

prelievo di istruzione e dati possono essere fatti insieme

Nel resto del corso considereremo un'architettura di Princeton

Memorie "cache" danno all'architettura di Princeton le prestazioni di un'architettura di Harvard

Struttura di un calcolatore

Schema a blocchi: vista a livello RTL

Seguiamo il percorso di un'istruzione: AC := AC + Mem<indirizzo>

1. **Prelievo istruzione (Instruction Fetch):**

   Trasferire PC a MAR

   Iniziare una sequenza di lettura da memoria

   Trasferire un dato (l'istruzione) da memoria ad IR

2. **Decodifica istruzione (Instruction Decode):**

   I bit di IR con il codice dell'istruzione sono gli ingressi dell'FSM di controllo

   Gli altri bit di IR rappresentano l'indirizzo dell'operando

Struttura di un calcolatore

Schema a blocchi: vista a livello RTL

Seguiamo il percorso di un'istruzione: AC := AC + Mem<indirizzo>

3. **Prelievo dell'operando (Operand Fetch):**

   Trasferire l'indirizzo dell'operando da IR a MAR

   Iniziare una sequenza di lettura da memoria

4. **Esecuzione dell'istruzione (Instruction Execute):**

   Il dato e' disponibile sul percorso di lettura

   Trasferire il dato all'ingresso dell'ALU

   Configurare l'ALU per eseguire una somma

   Trasferire il risultato da S ad AC

5. **Operazioni ausiliarie:**

   Incrementare PC per puntare alla prossima istruzione

Struttura di un calcolatore

Schema a blocchi: vista a livello RTL

Controllo: trasferire i dati da un registro all'altro attivando i segnali di controllo opportuni

Notazione per trasferimento tra registri (RTL)

Trasferimento tra registri

Bit di IR con codice operativo

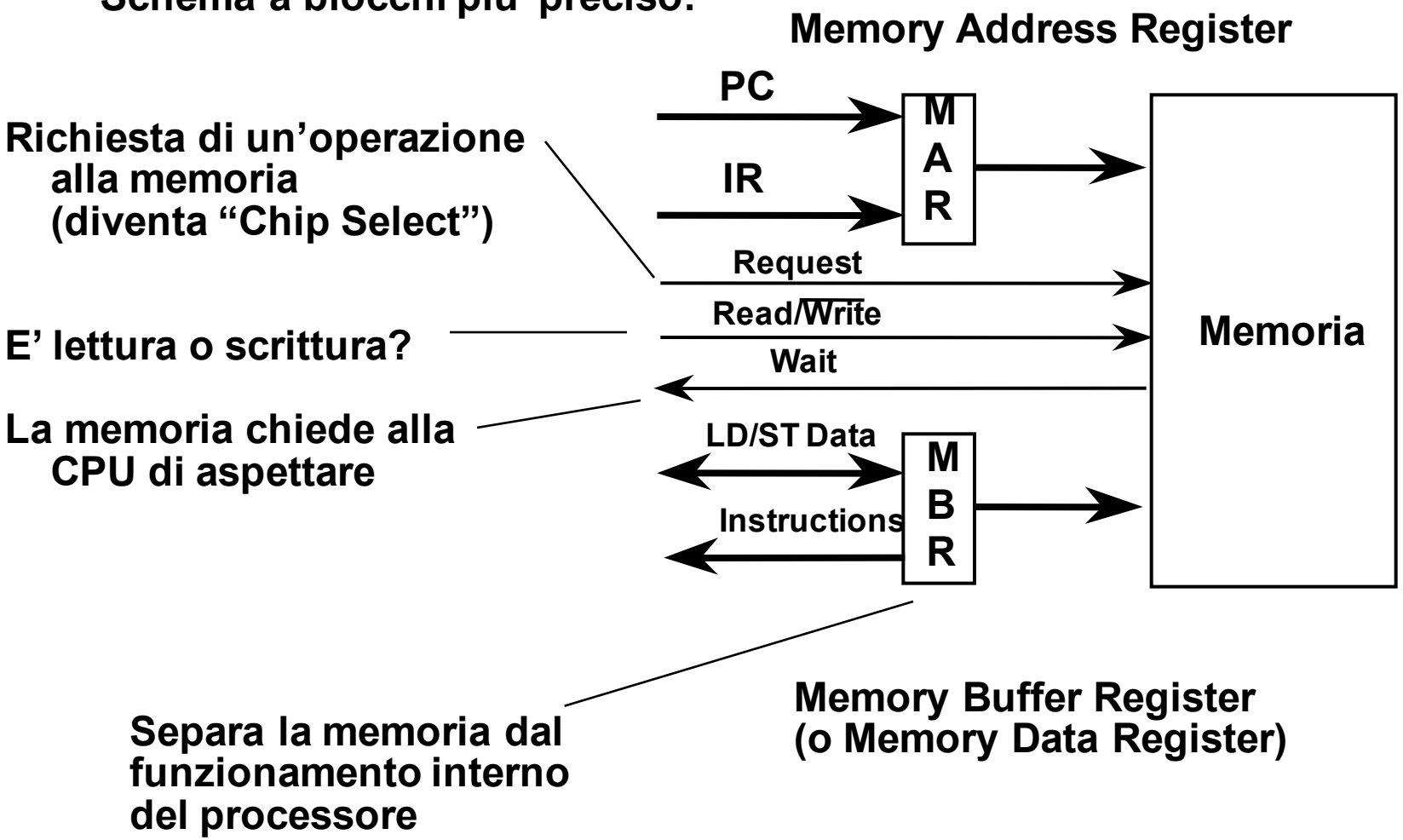
Ifetch:	**PC → MAR;**	**-- trasferire PC a MAR**
	Memory Read;	**-- attivare segnale lettura mem.**
	Memory → IR;	**-- caricare IR da memoria**
Instruction Decode:	**IF IR<op code> = ADD_FROM_MEMORY THEN**	
Instruction Execution:	**IR<addr> → MAR;**	**-- trasf. l'indir. operando a MAR**
	Memory Read;	**-- attivare segnale lettura mem.**
	Memory → ALU B;	**-- trasf. memoria ad ALU B**
	AC → ALU A;	**-- trasf. AC ad ALU A**
	ALU ADD;	**-- dire all'ALU di eseguire ADD**
	ALU S → AC;	**-- trasf. risultato ALU ad AC**
	PC+1;	**-- incrementare il PC**

Attivare segnale di controllo

Struttura di un calcolatore

Interfaccia con la memoria

Schema a blocchi piu' preciso:



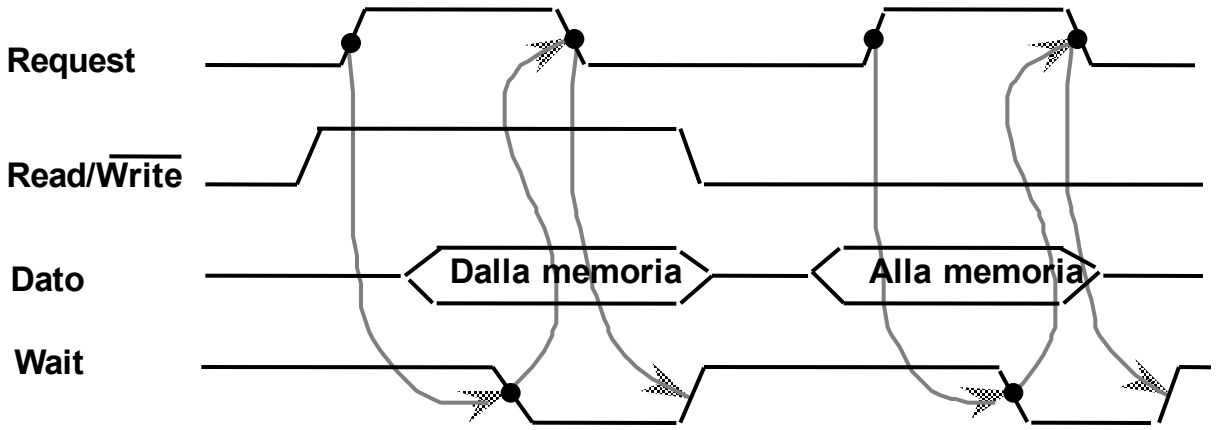
Struttura di un calcolatore

Interfaccia con la memoria

La CPU e la memoria non hanno un clock in comune

Seguono un ciclo asincrono con protocollo request/wait ($\overline{\text{ack}}$) a 4 fasi

1. **Attiva Request**
2. **Disattiva Wait**
3. **Disattiva Request**
4. **Attiva Wait**



Ciclo di lettura ***Ciclo di scrittura***

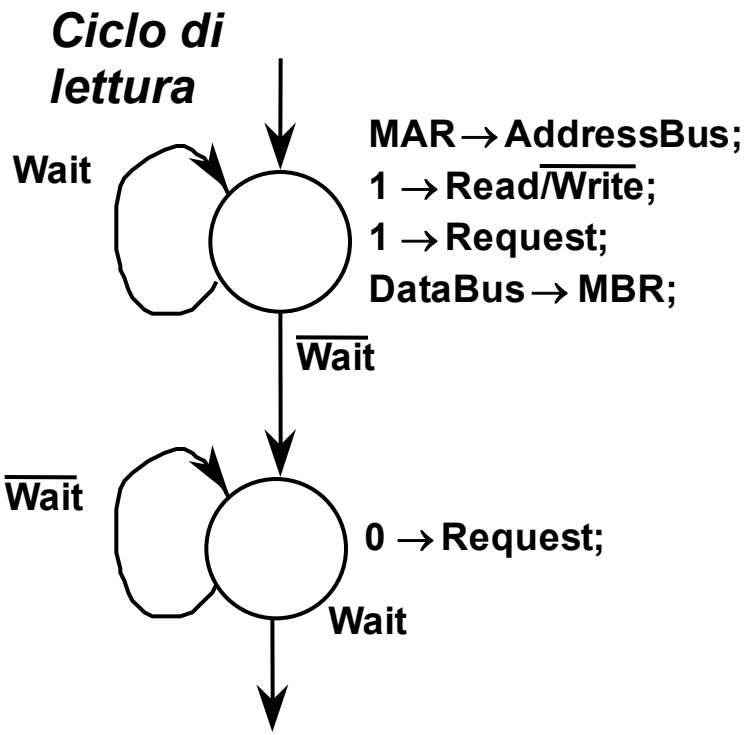
La CPU non puo' fare una nuova richiesta se Wait non e' attivo

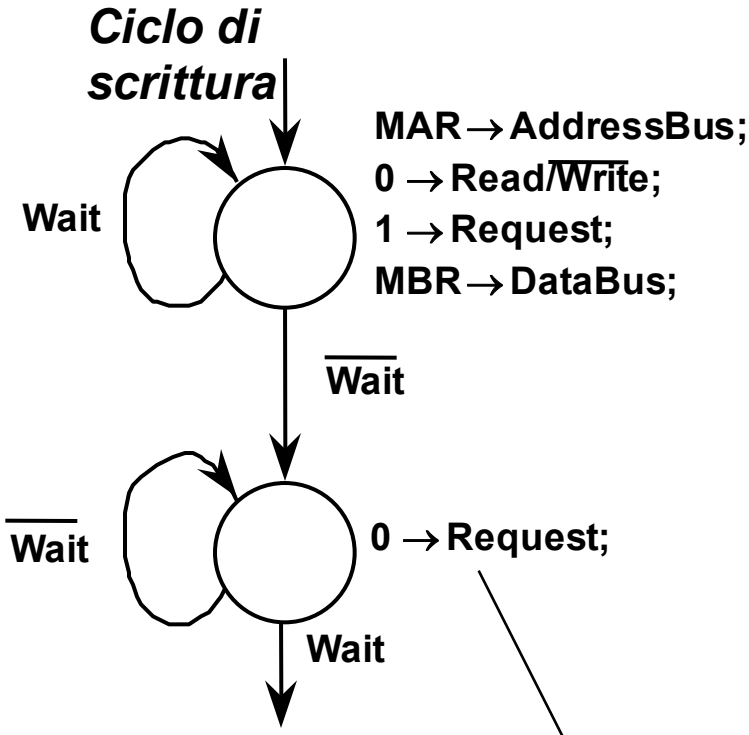
Fronte di discesa su Wait dice che il dato e' pronto (lettura) o e' stato scritto in memoria (scrittura)

Struttura di un calcolatore

Interfaccia con la memoria

Frammenti di diagramma degli stati per cicli di lettura e scrittura





Stato 1: pilotare bus indirizzi
attivare Request e lettura (R/W=1)
memorizzare dato in MBR

Stato 2: disattivare Request
restare in questo stato finche'
Wait e' riattivato

Convenzione normale:

Se un trasferimento tra registri NON va eseguito, non va menzionato nel diagramma

Struttura di un calcolatore

Interfaccia di ingresso/uscita (I/O)

I/O mappato in memoria (Memory-Mapped)

- **I dispositivi di I/O sono nello stesso spazio di indirizzi della memoria**
- **Registri di controllo gestiti come se fossero parole di memoria**
- **Lettura/scrittura su registri speciali per iniziare operazioni di I/O**

Polling

- **Programma controlla periodicamente se operazione di I/O e' finita**

Interruzione (interrupt)

- **Dispositivo segnala alla CPU quando operazione di I/O e' finita**
- **Software deve trasferire i dati dal dispositivo di I/O**
- **La CPU controlla se ci sono interruzioni prima di ogni fetch**
- **Salva PC e preleva la prossima istruzione da una locazione speciale ("vettore di interruzione")**
- **Tra le istruzioni c'e' "ritorno da interruzione"**

Organizzazione a bus

Comunicazioni tra registri

- Dirette (da punto a punto)
- Singolo bus condiviso
- Diversi bus per scopi diversi

Bisogna bilanciare bene la complessita' di UC/UO ed il grado di parallelismo offerto dall'hardware

Esempio:

Quattro registri di uso generale (General Purpose Register) che devono poter scambiare i propri contenuti

Il processore ha un'istruzione di scambio tra registri:

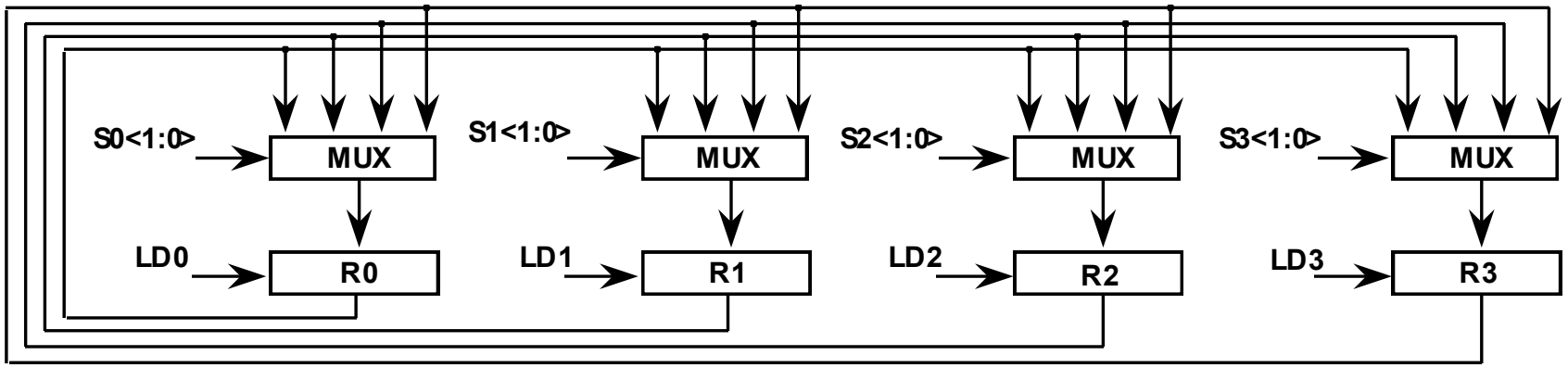
SWAP(Ri, Rj)

Ri $\rightarrow$ Rj;

Rj $\rightarrow$ Ri;

Schemi di connessione

Connessione diretta



Quattro registri connessi tramite 4 MUX 4:1 e connessioni dirette

- **Registri ad N bit attivi sul fronte e controllati dai segnali LDi**
- **N MUX 4:1 per ogni registro, controllati dai segnali Si<1:0>**

Schemi di connessione

Connessione diretta

Esempio:

Trasferimenti tra registri R1 → R0 ed R2 → R3

Operazioni di trasferimento tra registri:

01 → S0<1:0>;	**Abilita cammino tra R1 ed R0**
10 → S3<1:0>;	**Abilita cammino tra R2 ed R3**
1 → LD0;	**Attiva caricamento di R0**
1 → LD3;	**Attiva caricamento di R3**

Schemi di connessione

Connessione diretta

Quando sono attivati i segnali di controllo dei trasferimenti e quando sono eseguiti i trasferimenti stessi?

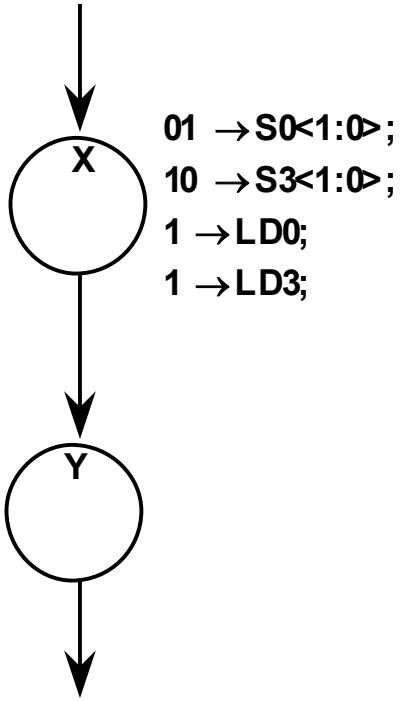


Diagramma a stati di una macchina di Moore

Entrando nello stato X:
- **Attiva i segnali di controllo dei multiplexer**
- **Le uscite di R1 arrivano agli ingressi di R0**
- **Le uscite di R2 arrivano agli ingressi di R3**

- **Attiva i segnali LDi**
- **Questi hanno effetto solo al fronte successivo del clock**

Entrando nello stato Y:
- **I segnali Ldi sono *sincroni* ed hanno effetto simultaneamente alla transizione di stato!**

Schemi di connessione

Connessione diretta

Realizzazione dello scambio tra registri

SWAP(R1, R2):

01 → S2<1:0>;
10 → S1<1:0>;

→ **Creazione delle connessioni**

1 → LD2;
1 → LD1;

→ **Lo scambio avviene alla prossima transizione di stato**

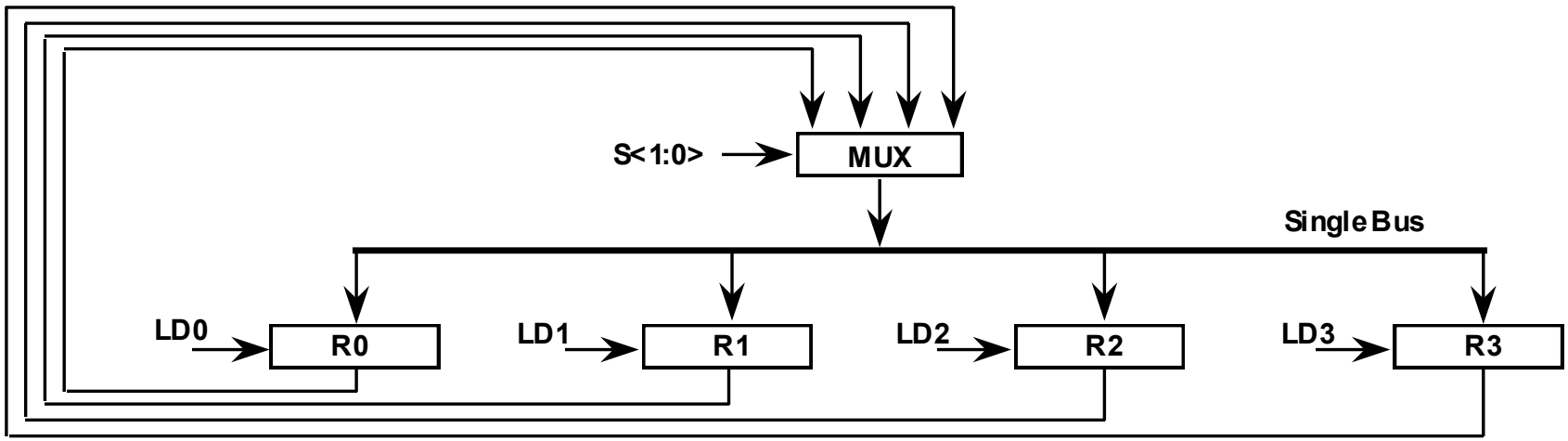
Vantaggi e svantaggi della connessione diretta:

+ possiamo trasferire un nuovo valore in tutti e quattro registri simultaneamente
+ lo scambio tra registri richiede un solo stato della FSM di controllo

- servono 5 porte logiche per ogni MUX 4:1
UO a 32 bit richiede 32 x 5 x 4 = 640 porte!
realizzazione molto costosa

Schemi di connessione

Connessione a bus singolo



- **Un solo MUX al posto di un MUX per registro**
- **Costa il 25% della soluzione precedente**
- **Un insieme *condiviso* di percorsi e' chiamato BUS**

Un bus singolo diventa una *risorsa critica*:
puo' essere usato per un solo trasferimento alla volta

Schemi di connessione

Connessione a bus singolo

Esempio: R1 → R0 ed R2 → R3

Stato X: **(R1 → R0)**

01 → S<1:0>;

1 → LD0;

Stato Y: **(R2 → R3)**

10 → S<1:0>;

1 → LD3;

L'UO non permette piu' trasferimenti simultanei!
Servono due stati dell'FSM di controllo per eseguire i due trasferimenti

Schemi di connessione

Connessione a bus singolo

Realizzazione dello scambio tra registri

Serve uno speciale registro temporaneo TEMP ("registro 4")
I MUX diventano 5:1 invece che 4:1

Stato X: (R1 → R4)

001 → S<2:0>;

1 → LD4;

Stato Y: (R2 → R1)

010 → S<2:0>;

1 → LD1;

Stato Z: (R4 → R2)

100 → S<2:0>

1 → LD2;

Servono tre stati invece di uno!

Inoltre servono un registro in piu'
e MUX con piu' ingressi

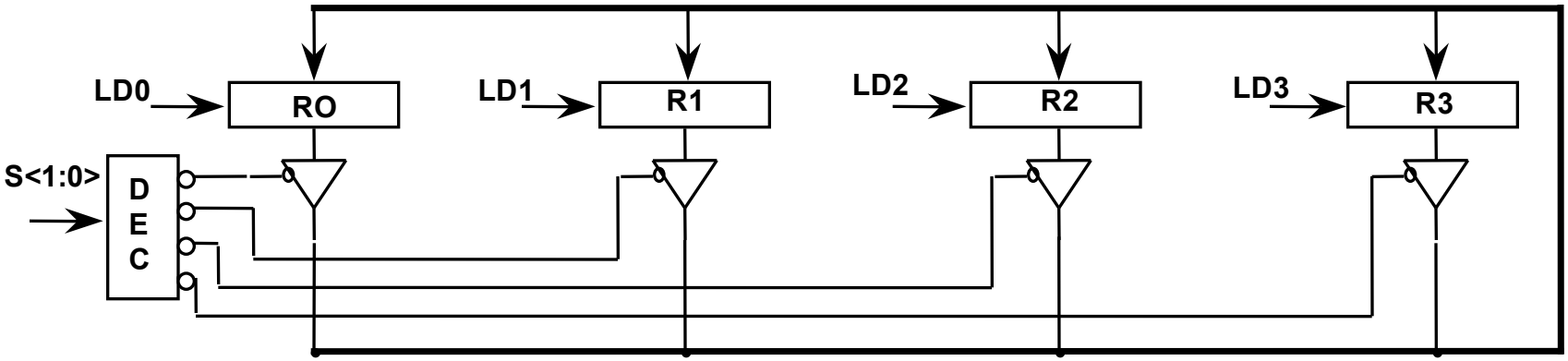
Sono necessari piu' stati della UC
perche' l'UO ha meno parallelismo

Scelte "ingegneristiche", basate su
quanto sono frequenti
i trasferimenti multipli simultanei!

Schemi di connessione

Alternative ai multiplexer

Usare buffer tri-state come meccanismo di interconnessione



Il contenuto di un solo registro puo' passare sul bus condiviso in un dato istante (ciclo di clock)

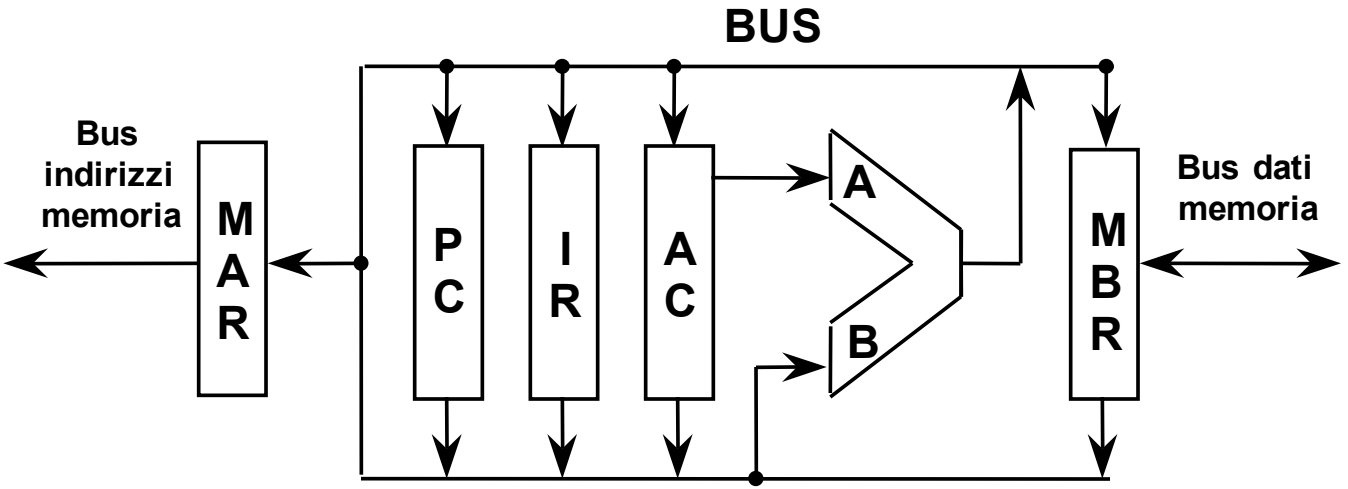
Schemi di connessione

Connessione a bus singolo

Le UO reali sono un compromesso tra questi due estremi

Diagramma di trasferimento tra registri

Progetto a bus singolo



Operazioni di trasferimento tra registri possibili:

PC → BUS
IR → BUS
AC → BUS
MBR → BUS
ALU Result → BUS

BUS → PC
BUS → IR
BUS → AC
BUS → MBR
BUS → ALU B
BUS → MAR

AC → ALU A
(fisso, o "hardwired")

Schemi di connessione

Connessione a bus singolo

Esempio di trasferimento tra registri per progetto a bus singolo

Esecuzione dell'istruzione "ADD Mem[X]"

Prelievo operando

Ciclo 1: IR<indirizzo operando> → BUS;
BUS → MAR;

Ciclo 2: Lettura memoria;
Databus → MBR;

Esecuzione somma

Ciclo 3: MBR → BUS;
BUS → ALU B;
AC → ALU A;
ADD;

Scrittura risultato

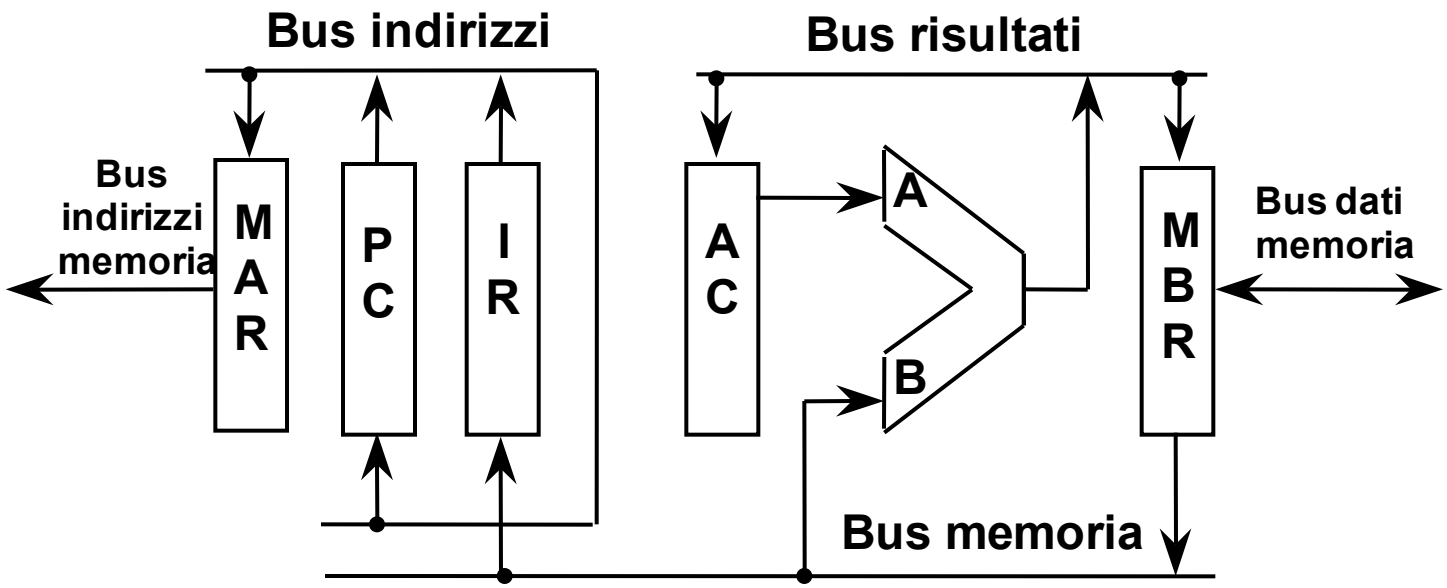
Ciclo 4: ALU Result → BUS;
BUS → AC;

C'e' bisogno di un registro per il risultato dell'ALU!

Schemi di connessione

Connessione a bus multipli

Progetto a tre bus permette piu' parallelismo:



Bus singolo sostituito da tre bus:

Bus memoria (MBUS)
Bus risultati (RBUS)
Bus indirizzi (ABUS)

MBUS ed RBUS non sono "veri bus" (singola sorgente)

Schemi di connessione

Connessione a bus multipli

Esecuzione dell'istruzione "ADD Mem[X]"

Prelievo operando

Ciclo 1:
IR<indirizzo operando> → ABUS;
ABUS → MAR;

Ciclo 2:
Lettura memoria;
Databus → MBR;

Esecuzione somma

Ciclo 3:
MBR → MBUS;
MBUS → ALU B;
AC → ALU A;
ADD;

Bastano tre cicli di clock invece di quattro!

Scrittura risultato
ALU Result → RBUS;
RBUS → AC;

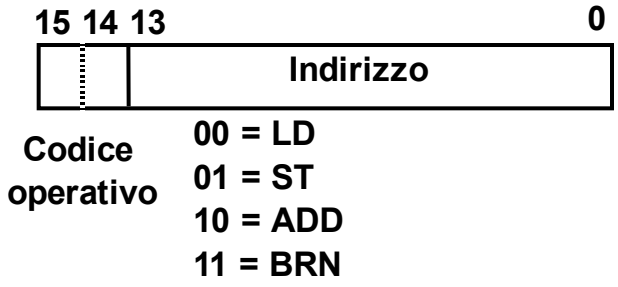
Vantaggio di ABUS separato:
Possiamo eseguire il prossimo PC → MAR durante l'esecuzione dell'istruzione precedente

Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

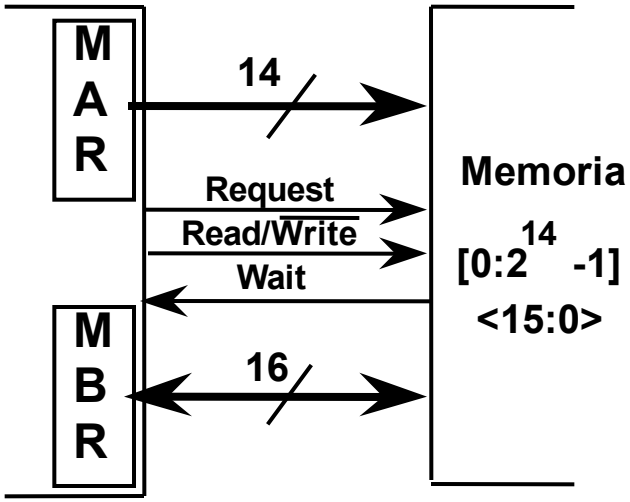
Specifica di un processore:

Formato dell'istruzione:



Lettura da memoria: Mem[XXX] → AC;
Scrittura in memoria: AC → Mem[XXX];
Somma da memoria: AC + Mem[XXX] → AC;
Salto se l'accumulatore e' negativo: AC < 0 ⇒ XXX → PC;

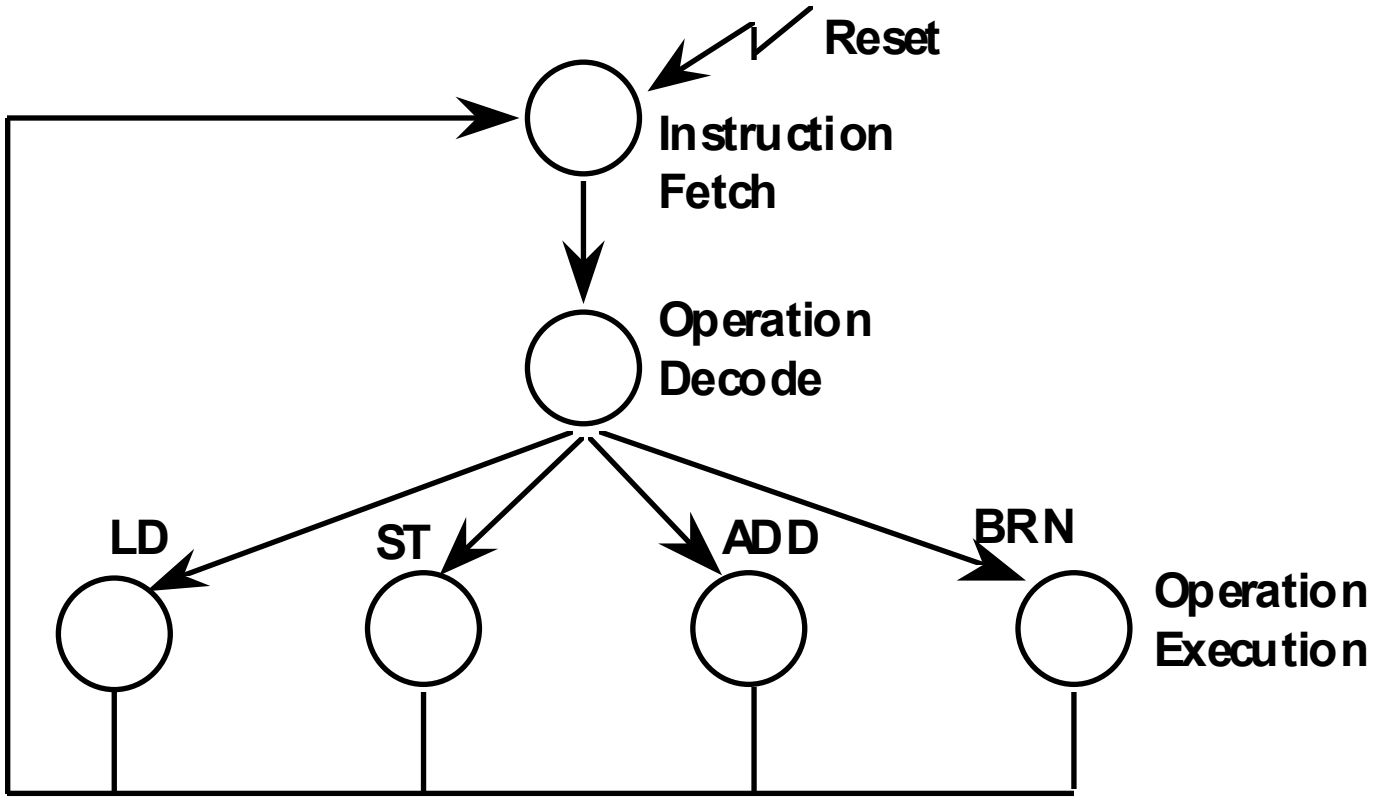
Interfaccia con la memoria:



Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Diagramma a stati iniziale:



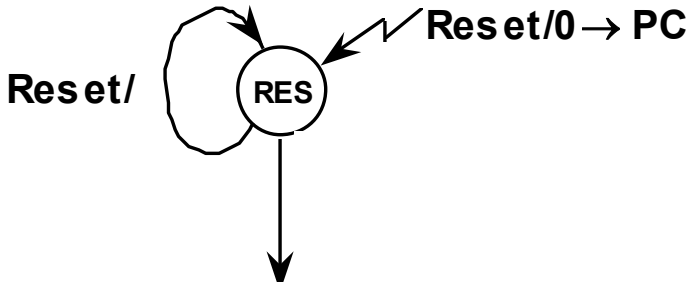
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Supponiamo macchina di Mealy sincrona:

Transizioni associate con archi anziche' con stati

Stato iniziale (stato 0) e sequenza di Instruction Fetch



All'inizializzazione:
- **azzerare il PC**
- **disattivare Mem Request**
- **la memoria attiva Wait**

,

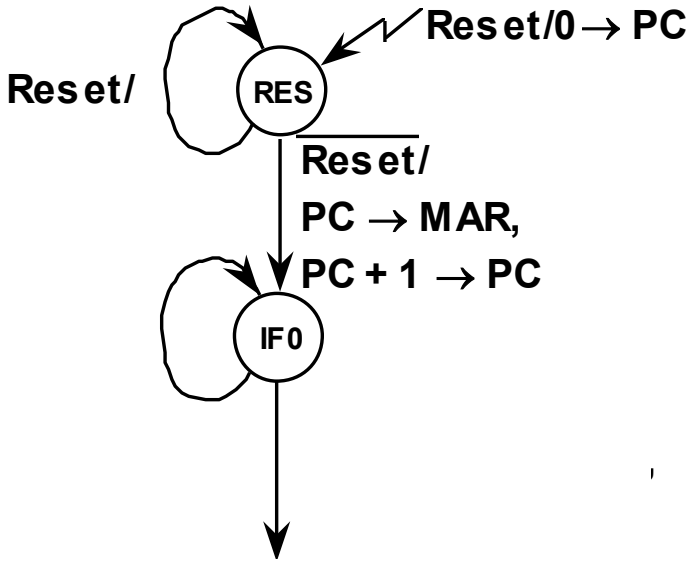
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Supponiamo macchina di Mealy sincrona:

Transizioni associate con archi anziche' con stati

Stato iniziale (stato 0) e sequenza di Instruction Fetch



All'inizializzazione:
- **azzerare il PC**
- **disattivare Mem Request**
- **la memoria attiva Wait**

Instruction Fetch:
- **attivare richiesta di lettura**
- **handshake a 4 fasi su Wait**

Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

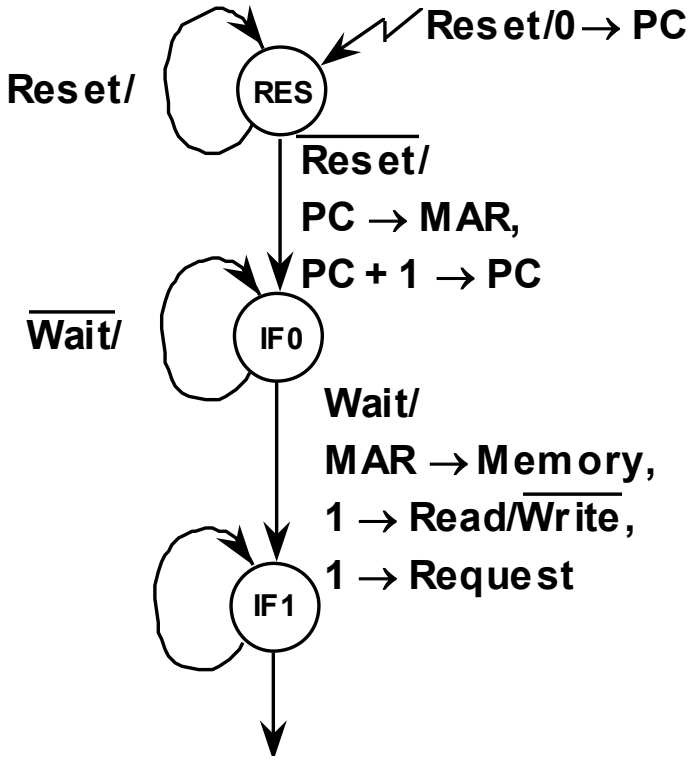
Supponiamo macchina di Mealy sincrona:

Transizioni associate con archi anziche' con stati

Stato iniziale (stato 0) e sequenza di Instruction Fetch

All'inizializzazione:
- **azzerare il PC**
- **disattivare Mem Request**
- **la memoria attiva Wait**

Instruction Fetch:
- **attivare richiesta di lettura**
- **handshake a 4 fasi su Wait**



Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Supponiamo macchina di Mealy sincrona:

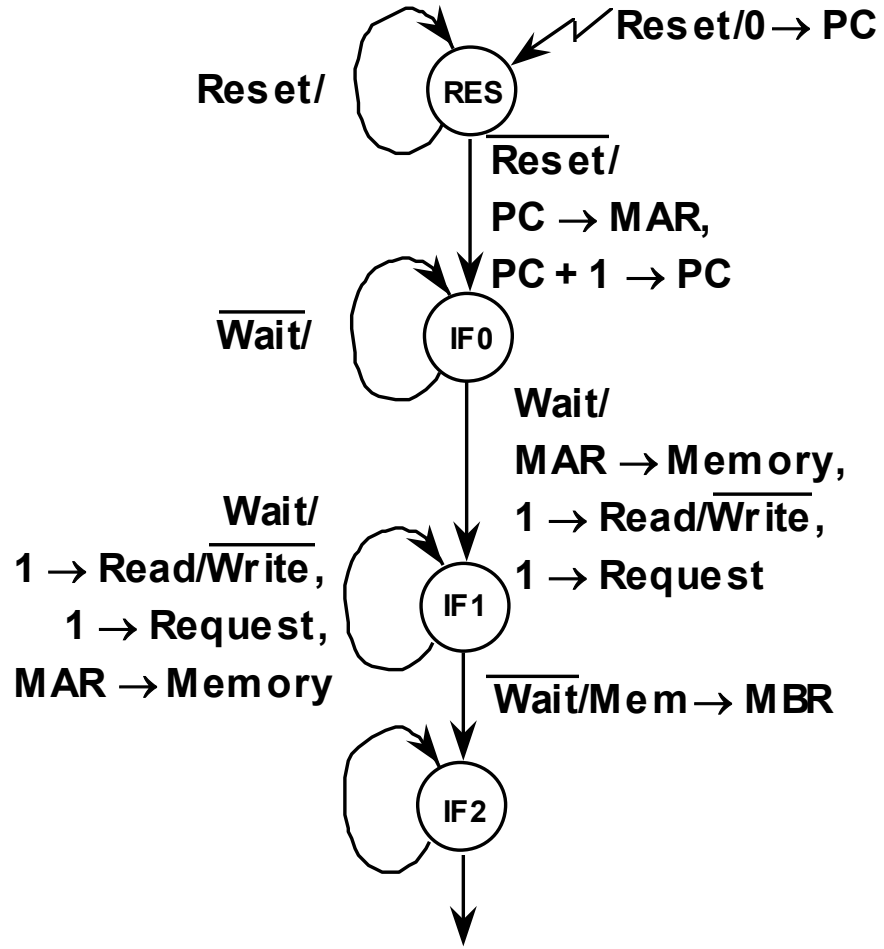
Transizioni associate con archi anziche' con stati

Stato iniziale (stato 0) e sequenza di Instruction Fetch

All'inizializzazione:
- **azzerare il PC**
- **disattivare Mem Request**
- **la memoria attiva Wait**

Instruction Fetch:
- **attivare richiesta di lettura**
- **handshake a 4 fasi su Wait**

Nota: per ora i bus usati per realizzare i trasferimenti non vengono menzionati!



Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Supponiamo macchina di Mealy sincrona:

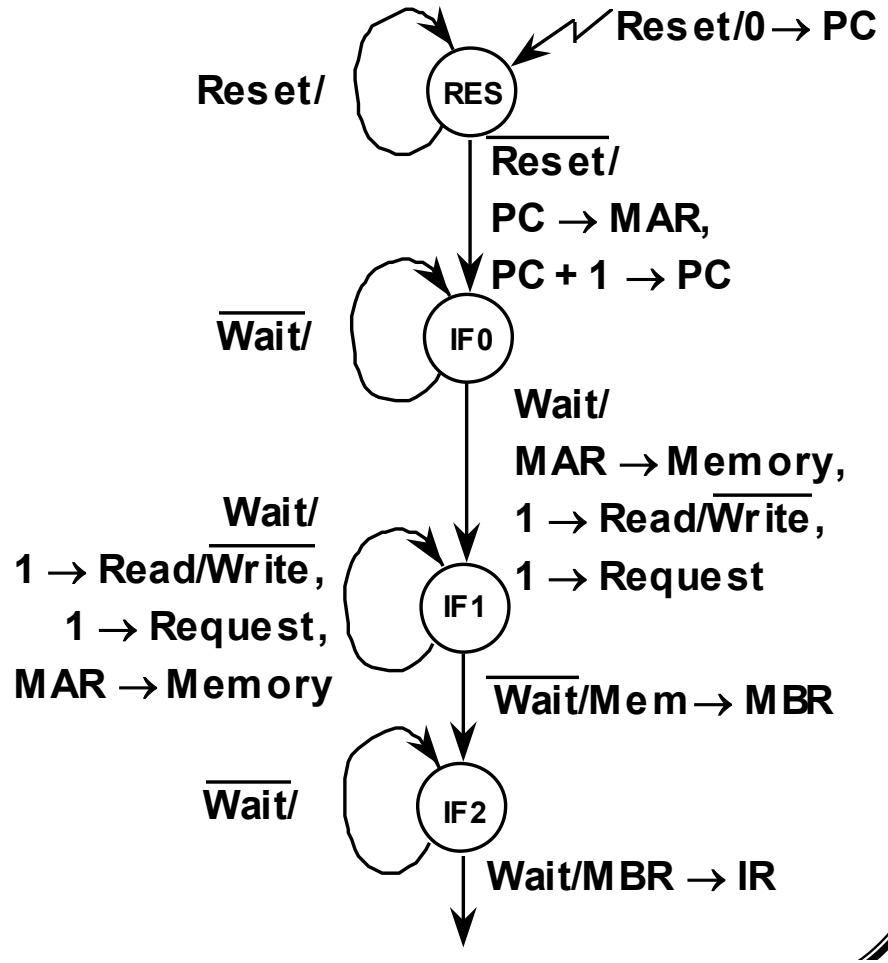
Transizioni associate con archi anziche' con stati

Stato iniziale (stato 0) e sequenza di Instruction Fetch

All'inizializzazione:
- **azzerare il PC**
- **disattivare Mem Request**
- **la memoria attiva Wait**

Instruction Fetch:
- **attivare richiesta di lettura**
- **handshake a 4 fasi su Wait**

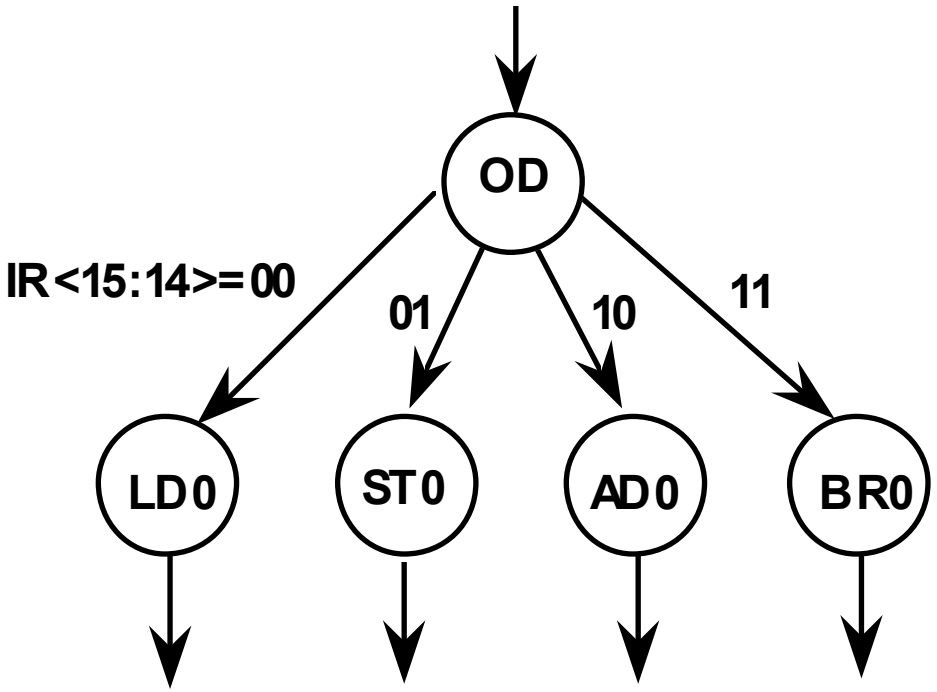
Nota: per ora i bus usati per realizzare i trasferimenti non vengono menzionati!



Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Stato di decodifica dell'istruzione



Quattro stati futuri a seconda dei bit del codice operativo

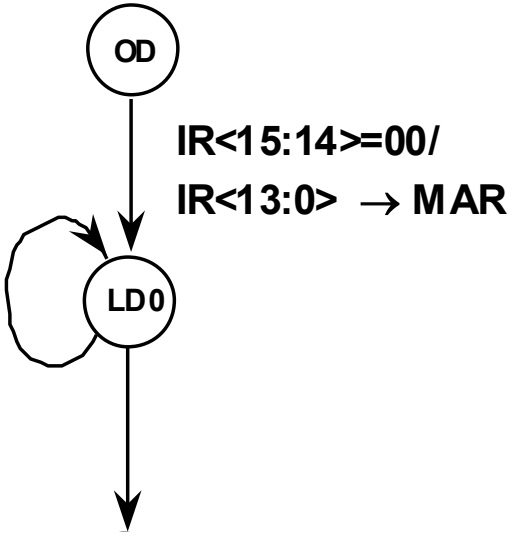
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di lettura da memoria

Come Instruction Fetch,
ma l'indirizzo dell'operando
arriva da IR ed il dato va
caricato in AC



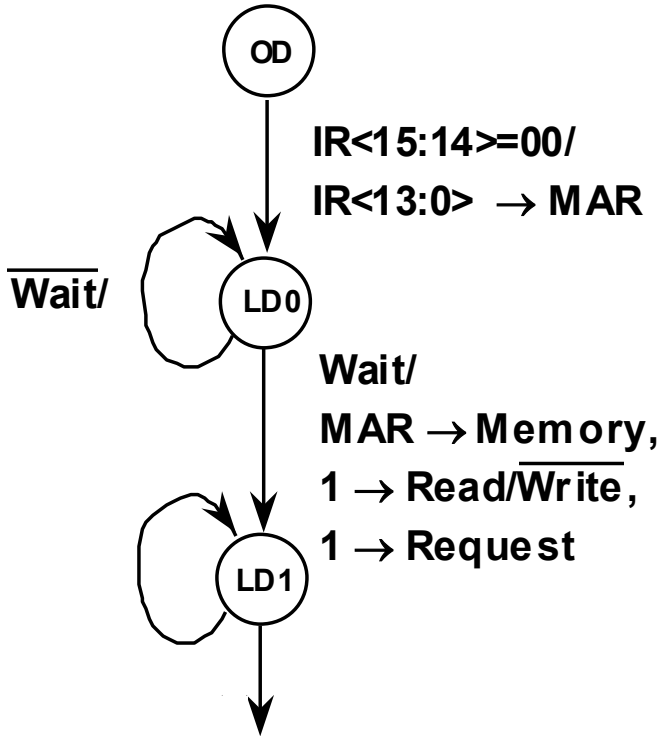
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di lettura da memoria

Come Instruction Fetch, ma l'indirizzo dell'operando arriva da IR ed il dato va caricato in AC



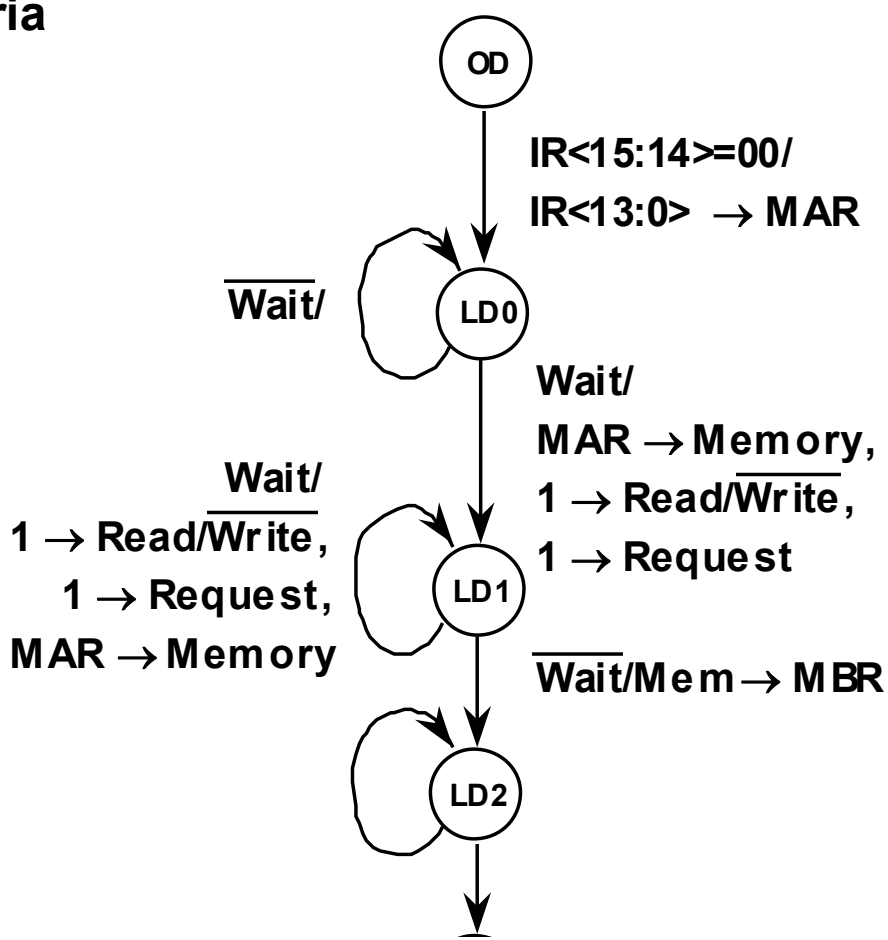
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di lettura da memoria

Come Instruction Fetch, ma l'indirizzo dell'operando arriva da IR ed il dato va caricato in AC



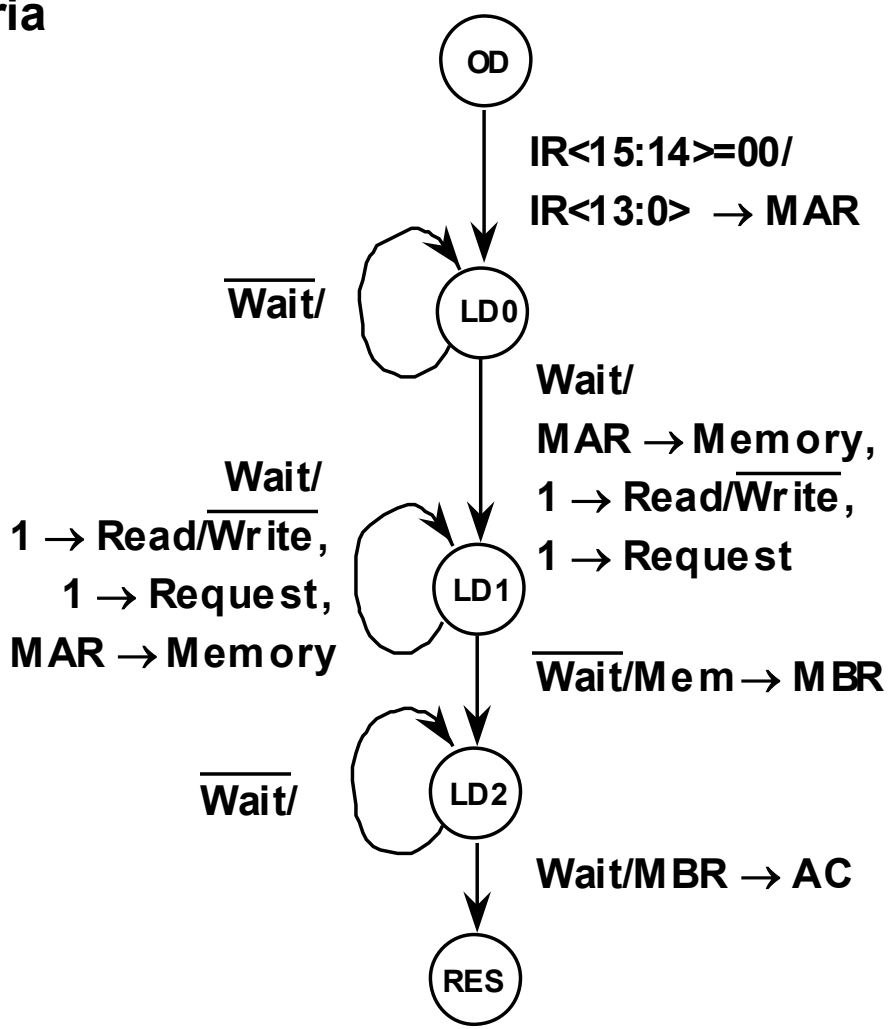
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di lettura da memoria

Come Instruction Fetch,
ma l'indirizzo dell'operando
arriva da IR ed il dato va
caricato in AC

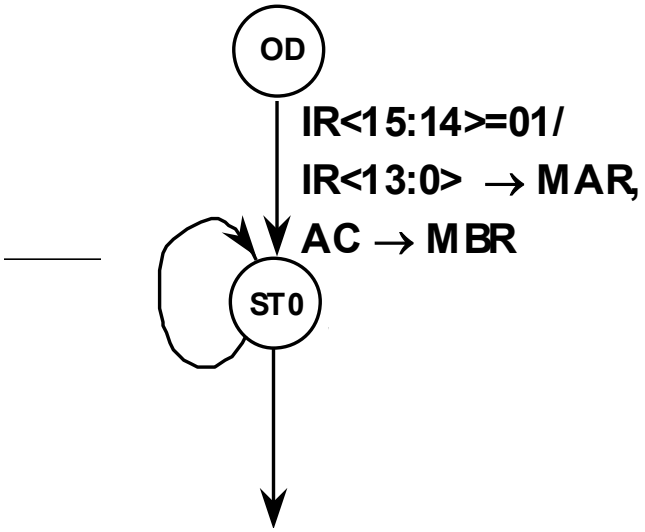


Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di scrittura in memoria

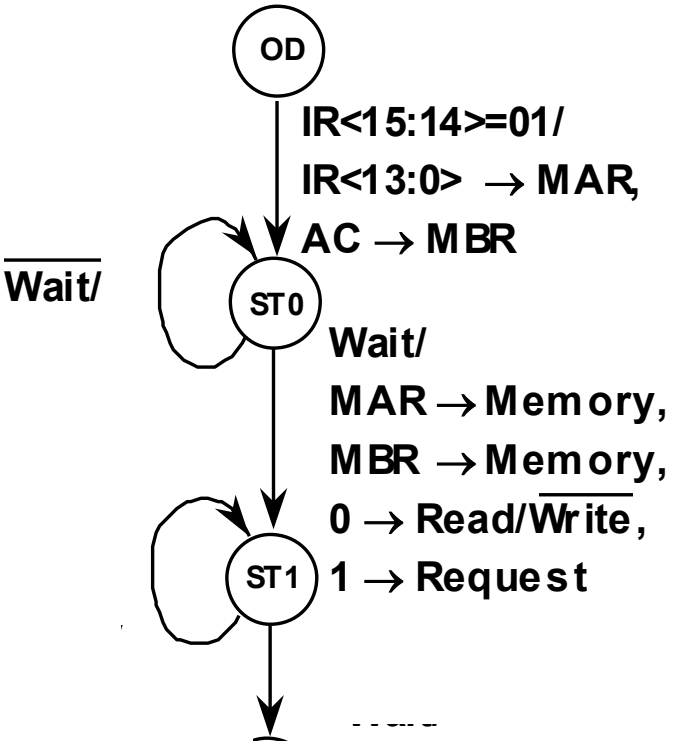


Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di scrittura in memoria

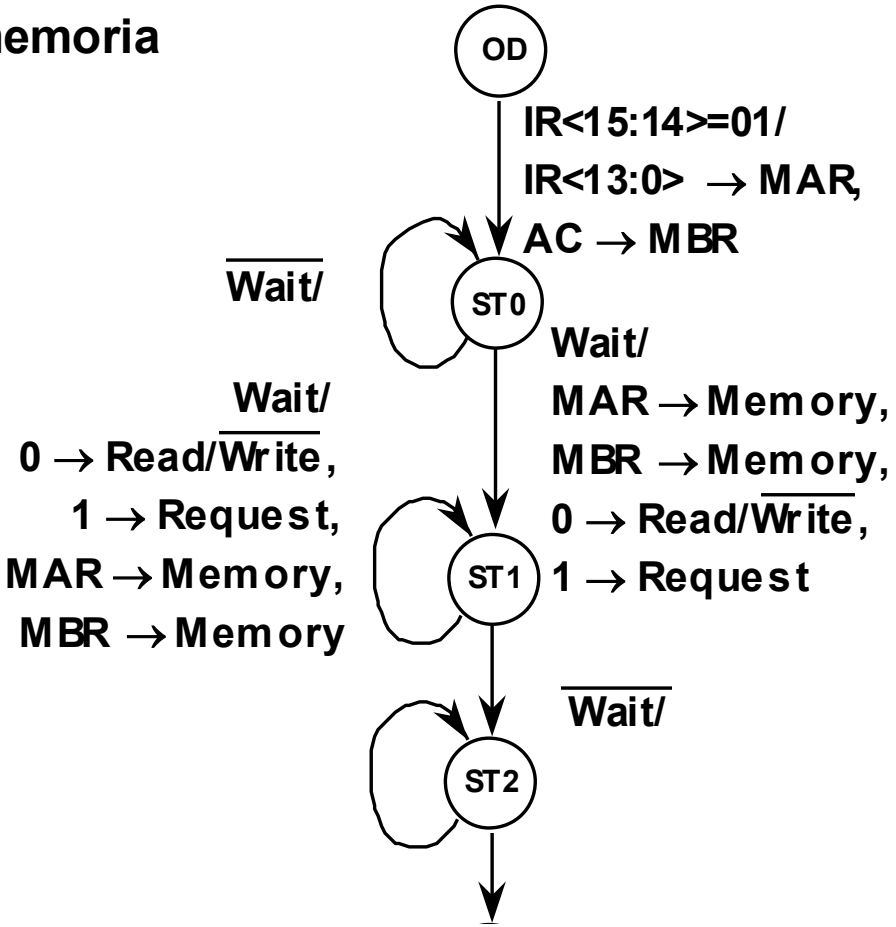


Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di scrittura in memoria

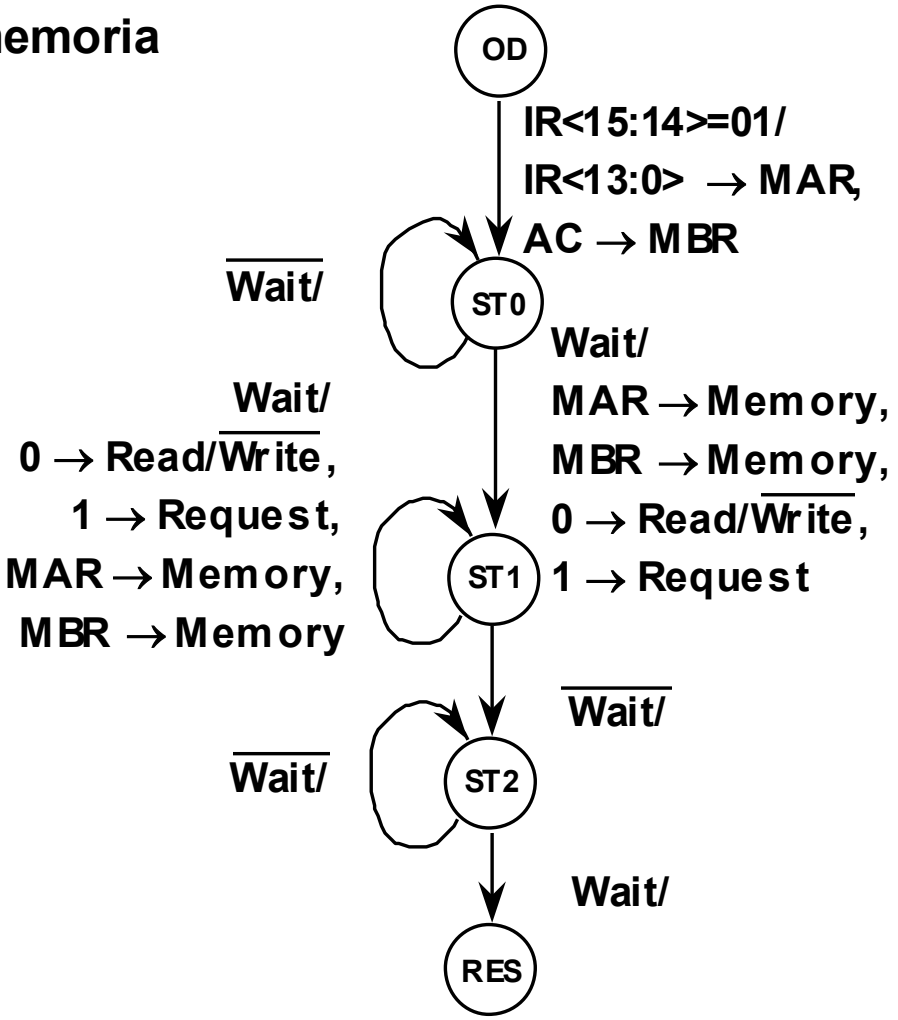


Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di scrittura in memoria



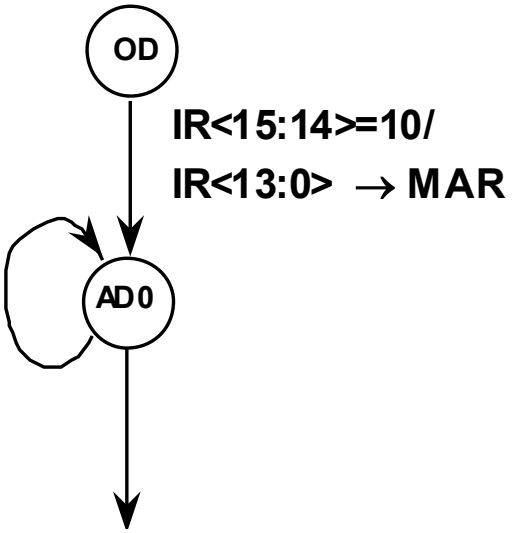
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di somma

Simile alla lettura, ma somma MBR ad AC invece di trasferire MBR in AC



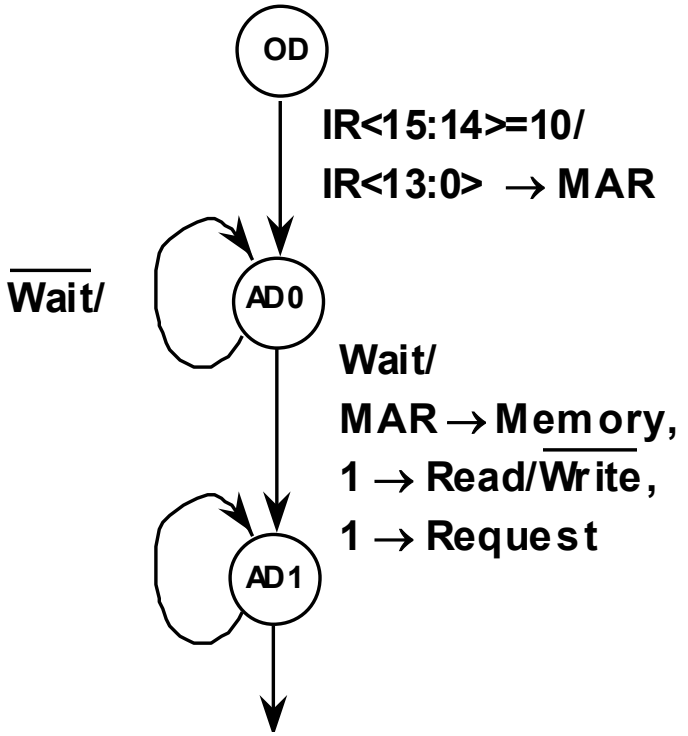
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di somma

Simile alla lettura, ma somma MBR ad AC invece di trasferire MBR in AC



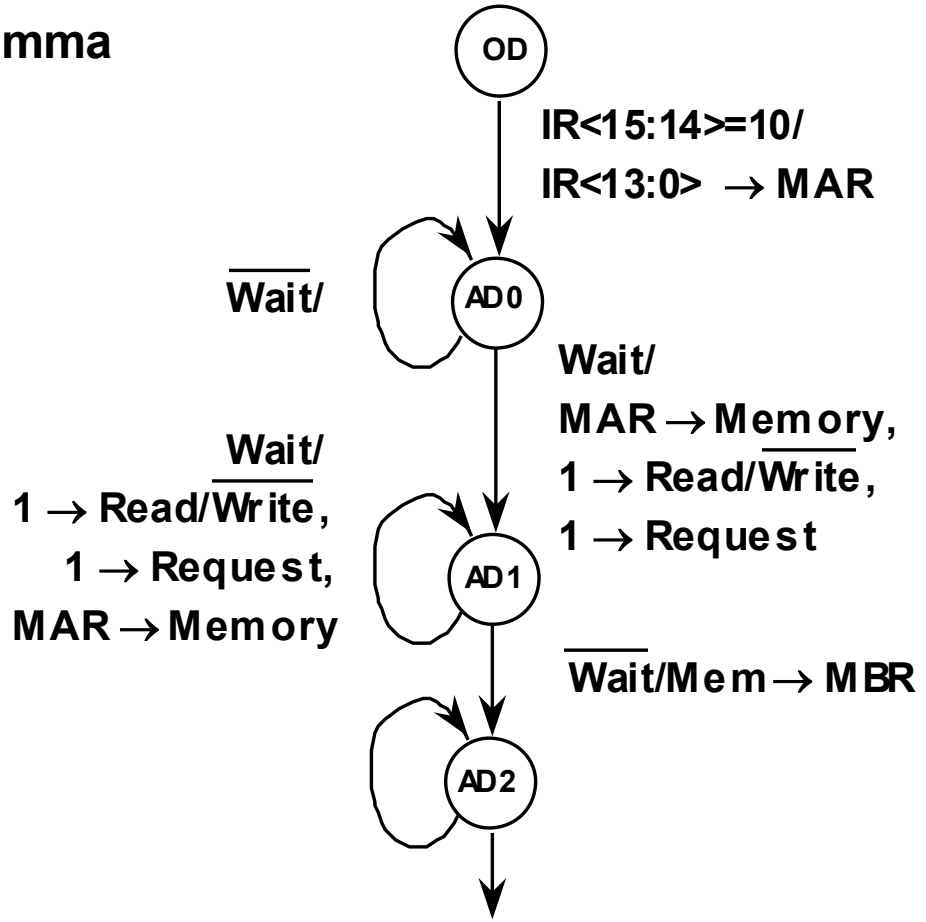
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di somma

Simile alla lettura, ma somma MBR ad AC invece di trasferire MBR in AC



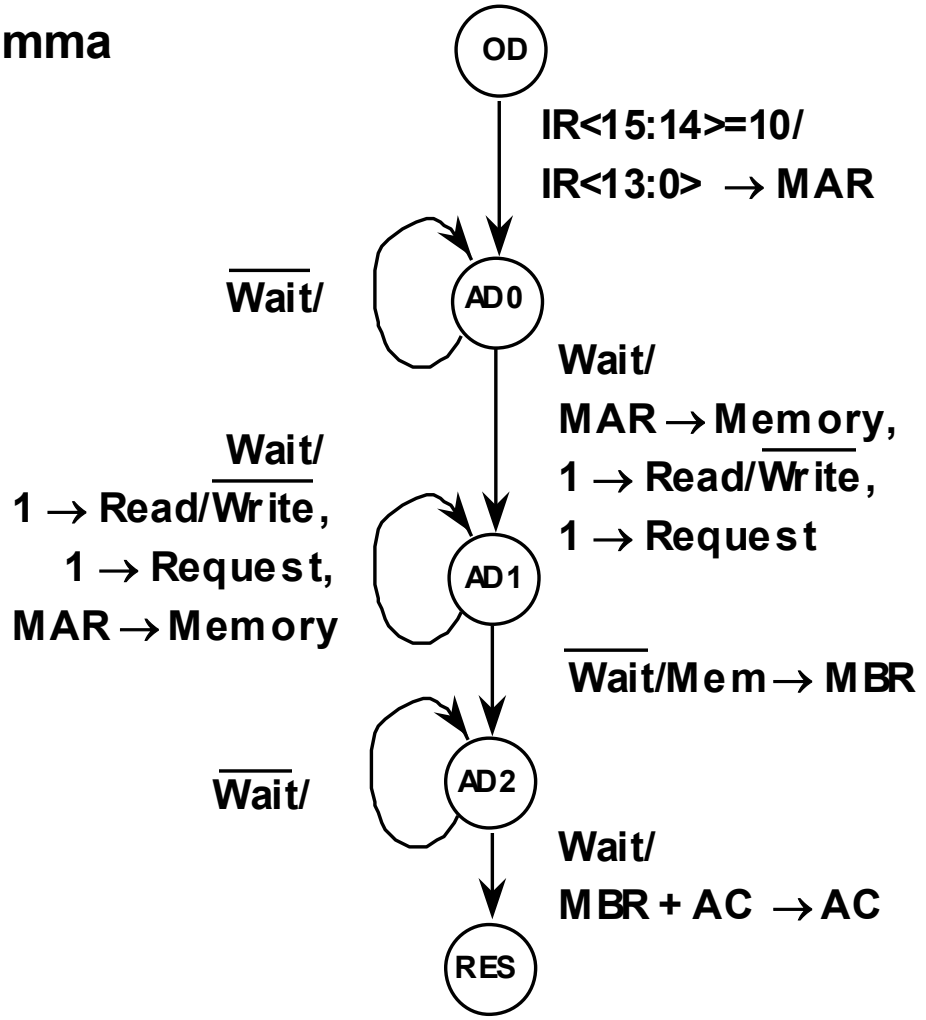
Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di somma

Simile alla lettura, ma somma MBR ad AC invece di trasferire MBR in AC

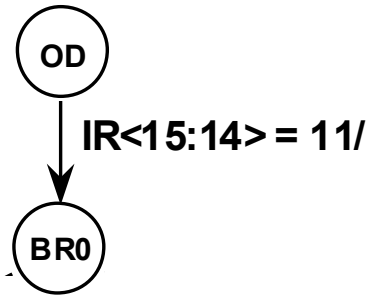


Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di salto

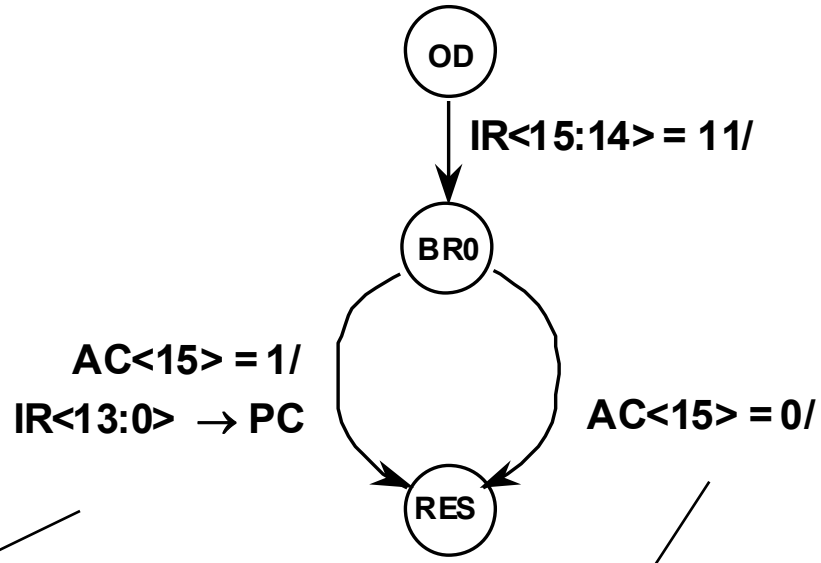


Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Sequenze di esecuzione

Sequenza di salto



Caricare il campo "indirizzo operando" nel PC se AC < 0

Altrimenti, non fare nulla

Macchine a Stati Finiti per CPU semplici

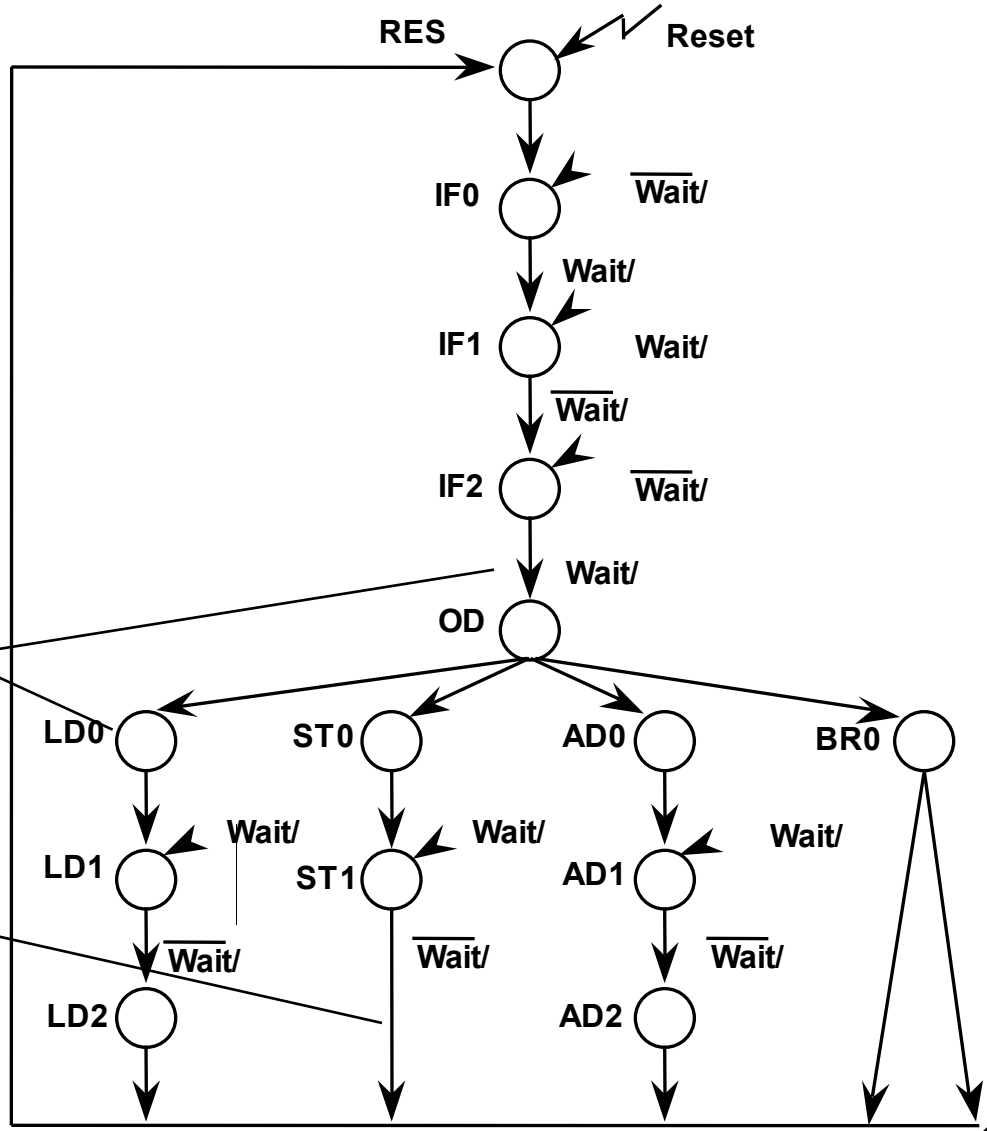
Progetto del diagramma a stati e dell'Unita' Operativa

Diagramma a stati completo

Si possono semplificare i cicli di attesa, eliminando alcuni stati di attesa

A questo punto, Wait deve essere attivo, quindi perche' eseguire un ciclo su $\overline{\text{Wait}}$?

Perche' eseguire un ciclo su $\overline{\text{Wait}}$ se ci risincronizzeremo comunque in IF0?



Macchine a Stati Finiti per CPU semplici

Progetto del diagramma a stati e dell'Unita' Operativa

Ingressi ed uscite della Macchina a Stati (fino a questo punto):

Ingressi:

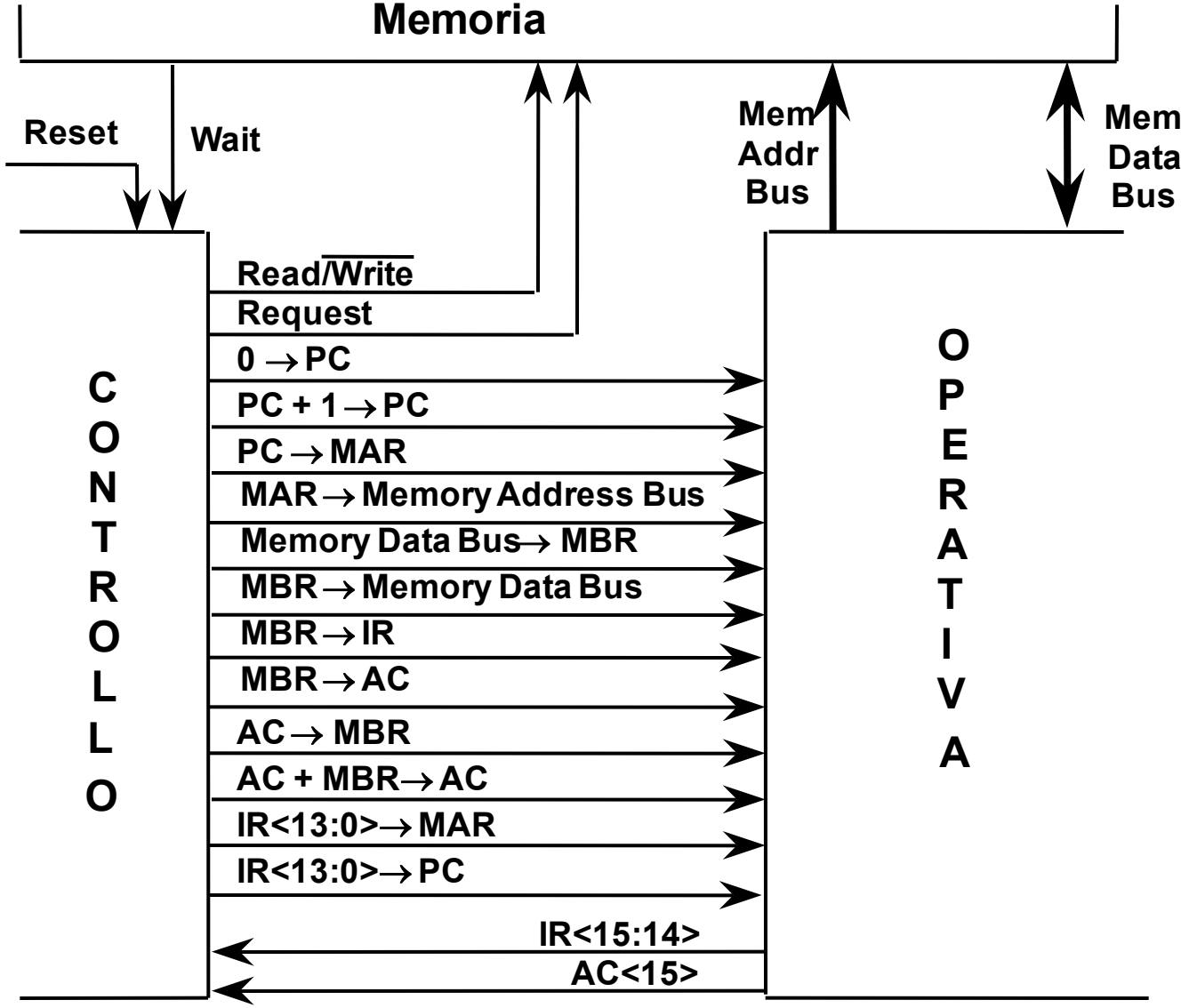
- Reset
- Wait
- IR<15:14>
- AC<15>

Uscite:

- 0 → PC
- PC + 1 → PC
- PC → MAR
- MAR → Memory Address Bus
- Memory Data Bus → MBR
- MBR → Memory Data Bus
- MBR → IR
- MBR → AC
- AC → MBR
- AC + MBR → AC
- IR<13:0> → MAR
- IR<13:0> → PC
- 1 → Read/Write
- 0 → Read/Write
- 1 → Request

Macchine a Stati Finiti per CPU semplici

Percorso dei segnali di controllo nel processore

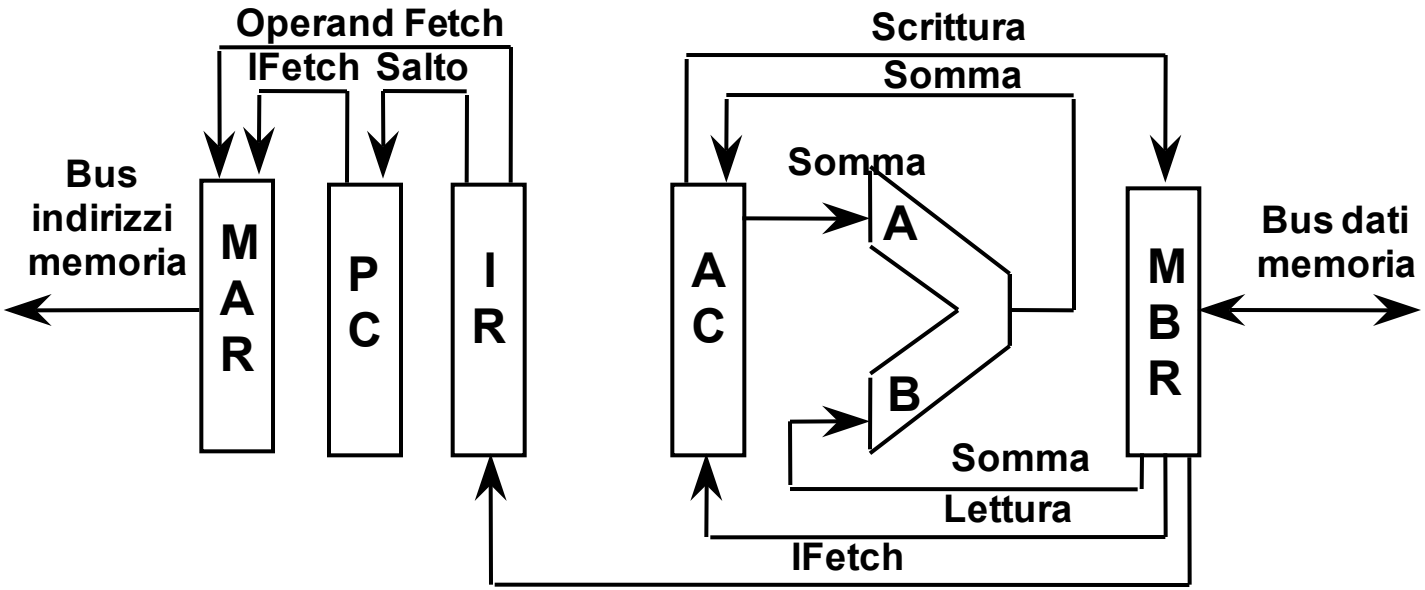


Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa

La specifica finora e' indipendente dalla strategia di connessione

Trasferimenti necessari:



Questo (supposto finora) e' lo schema di interconnessione punto a punto

Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa

Notiamo che prelievo istruzione (IFetch) e prelievo operando (Operand Fetch) non avvengono mai simultaneamente

Quindi i trasferimenti che coinvolgono IR, PC, e MAR possono essere realizzati da un solo bus (bus indirizzi, ABUS)

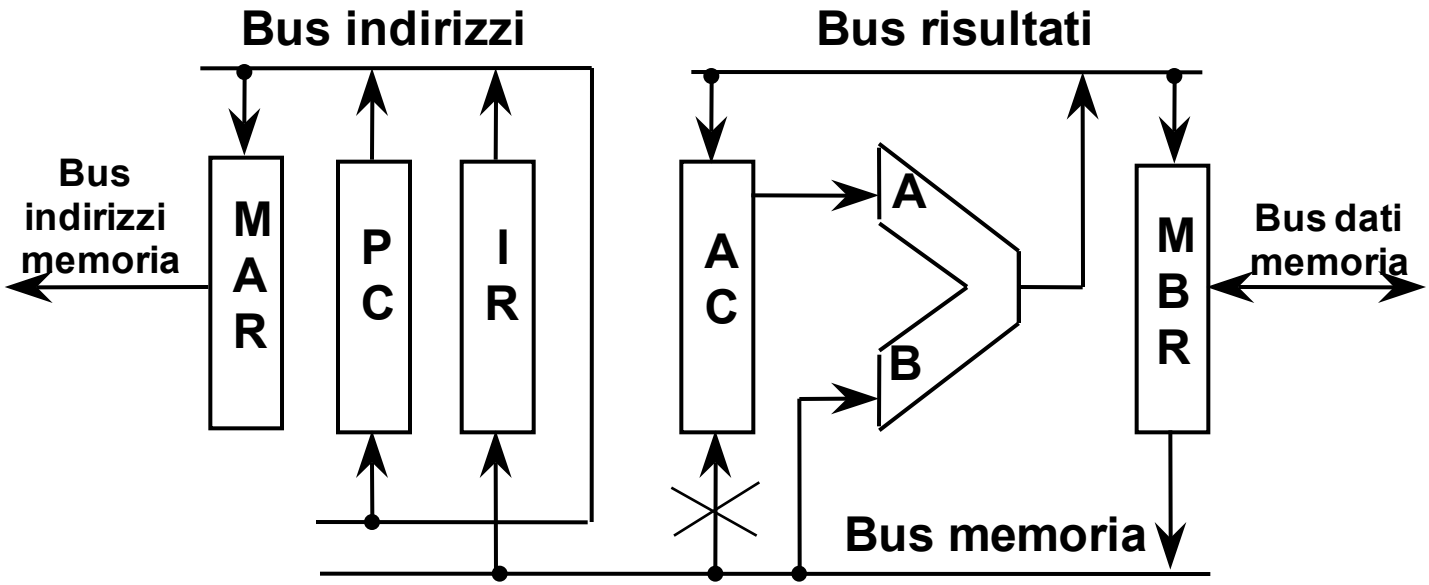
Combiniamo le connessioni di MBR, IR, ALU B, ed AC (bus memoria, MBUS)

Combiniamo le connessioni di ALU, AC, ed MBR (bus risultati, RBUS)

Architettura a tre bus:
AC + MBR $\rightarrow$ AC puo' essere eseguito in un solo stato

Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa



AC ha due ingressi, RBUS ed MBUS
(gli altri registri, eccetto MBR hanno un solo ingresso)

Un registro a due ingressi ("dual port") e' piu' complicato (MUX)

Un'idea migliore: riutilizzare le connessioni quando e' possibile
il trasferimento MBR → AC e' realizzato dall'operazione PASS B della ALU

Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa

Realizzazione in dettaglio dei trasferimenti tra registri

I passi elementari delle operazioni di controllo sono detti *microoperazioni*

Un singolo trasferimento tra registri puo' corrispondere a molte microoperazioni

Alcune operazioni sono realizzate direttamente dalle unita' funzionali:

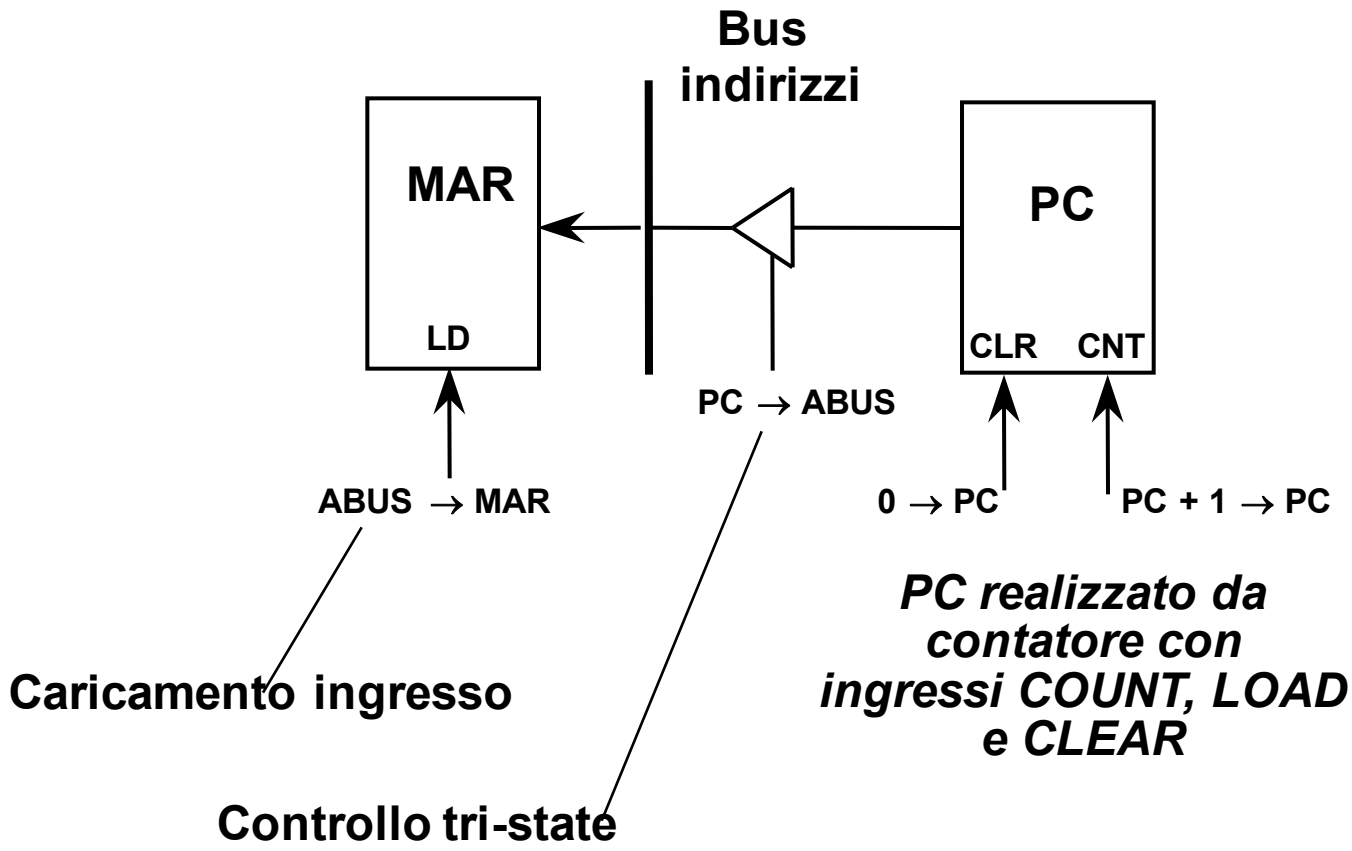
p.es., ADD, Pass B, 0 $\rightarrow$ PC, PC + 1 $\rightarrow$ PC

Altre operazioni richiedono molte microoperazioni:

p.es., PC $\rightarrow$ MAR realizzata come
PC $\rightarrow$ ABUS ed ABUS $\rightarrow$ MAR

Macchine a Stati Finiti per CPU semplici

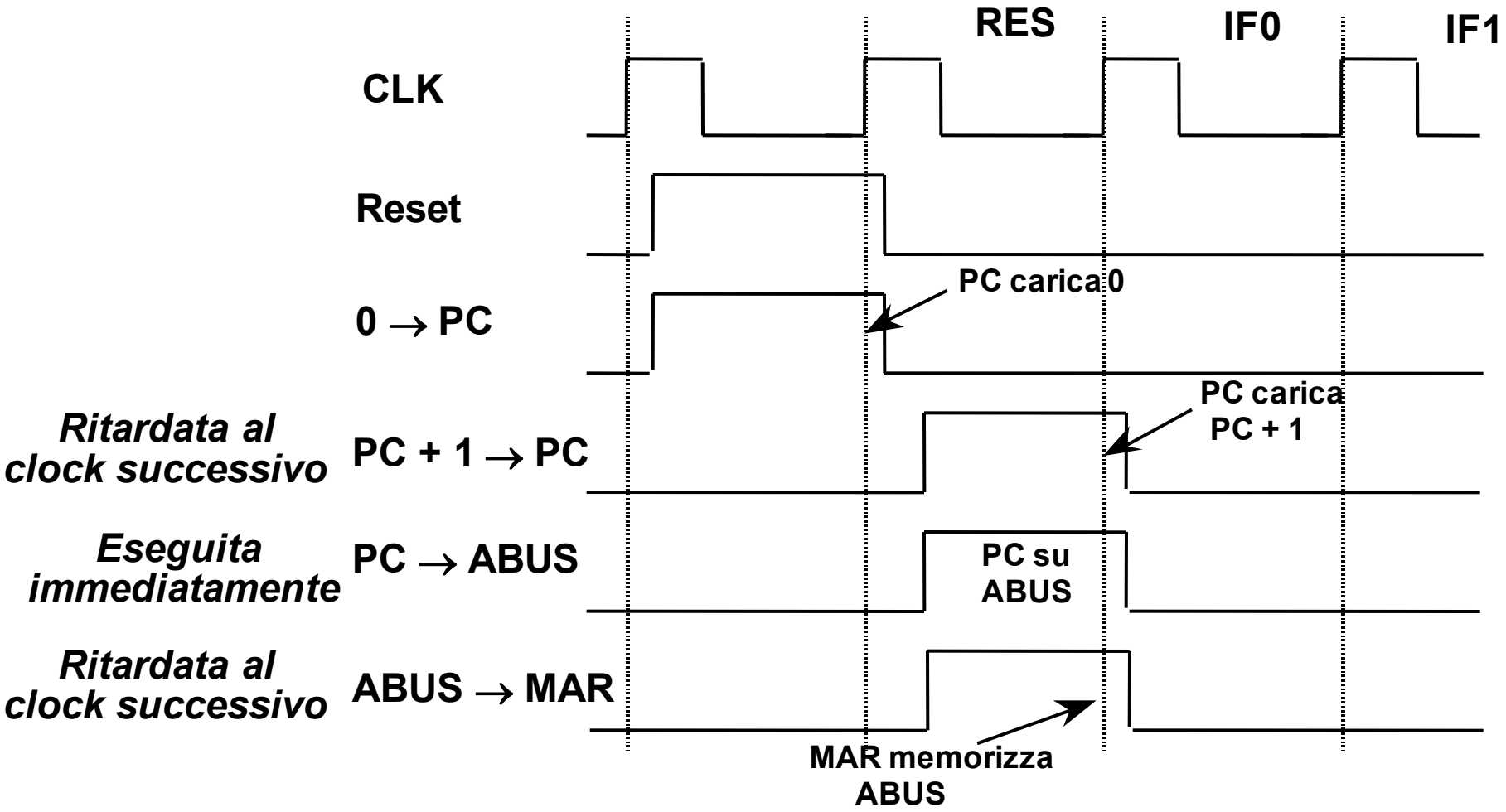
Trasformazione nel controllo dell'Unita' Operativa



Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa

Tempistica dele transizioni di stato e delle microoperazioni



Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa

Relazione fra trasferimenti fra registri e microoperazioni

Trasferimento tra registri	Microoperazioni
0 → PC	0 → PC (ritardata);
PC + 1 → PC	PC + 1 → PC (ritardata);
PC → MAR	PC → ABUS (immediata),
	ABUS → MAR (ritardata);
MAR → Bus Indirizzi	MAR → Bus indirizzi (immediata);
Data Bus → MBR	Bus dati → MBR (ritardata);
MBR → Bus dati	MBR → Bus dati (immediata);
MBR → IR	MBR → ABUS (immediata),
	ABUS → IR (ritardata);
MBR → AC	MBR → MBUS (immediata),
	MBUS → ALU B (immediata),
	ALU PASS B (immediata),
	Risultato ALU → RBUS (immediata),
	RBUS → AC (ritardata);

Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa

Relazione fra trasferimenti fra registri e microoperazioni

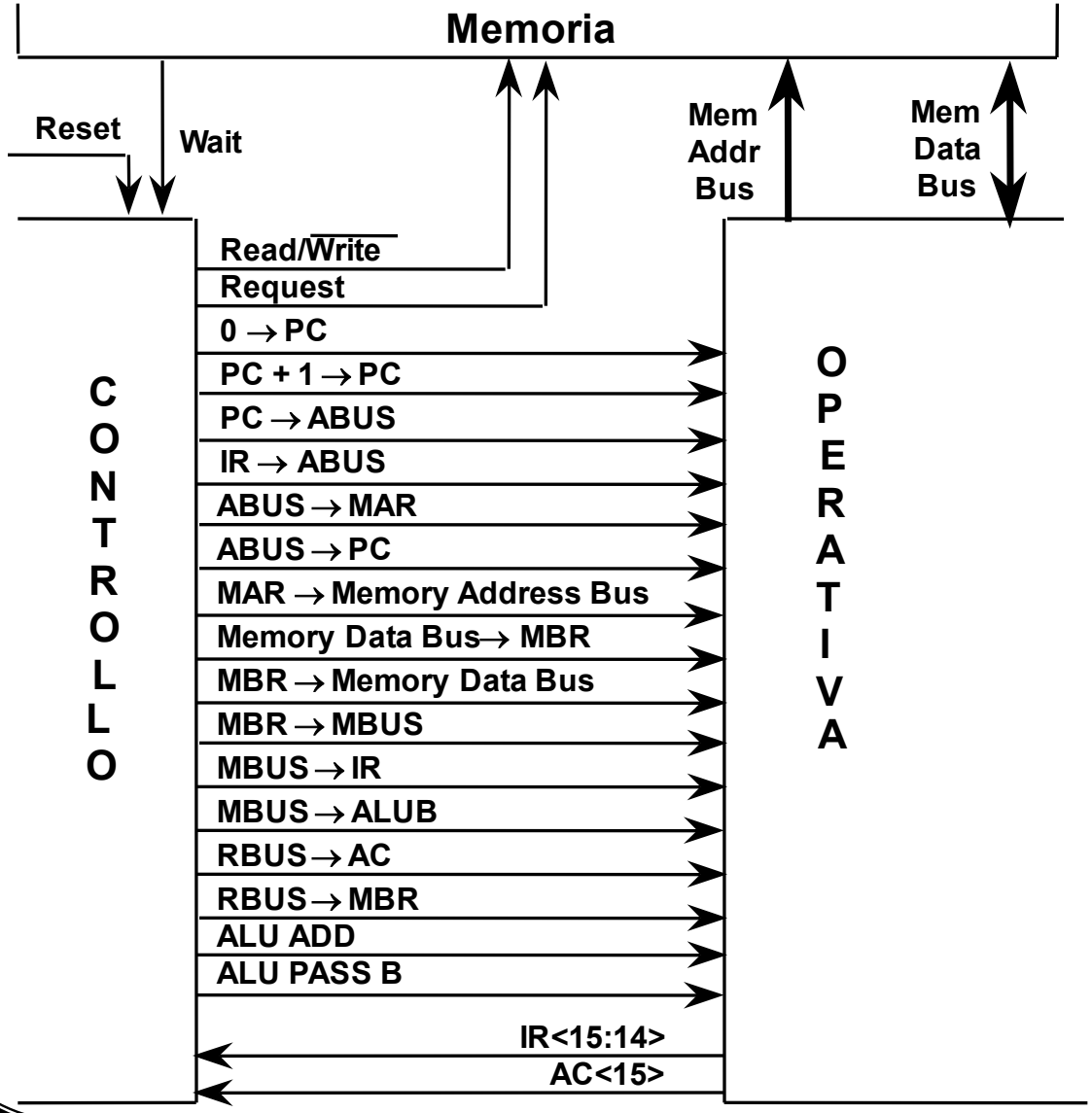
Trasferimento tra registri	Microoperazioni
AC → MBR	AC → RBUS (immediata), RBUS → MBR (ritardata);
AC + MBR → AC	AC → ALU A (immediata), MBR → MBUS (immediata), MBUS → ALU B (immediata), ALU ADD (immediata), Risultato ALU → RBUS (immediata), RBUS → AC (ritardata);
IR<13:0> → MAR	IR → ABUS (immediata), ABUS → MAR (ritardata);
IR<13:0> → PC	IR → ABUS (immediata), ABUS → PC (ritardata);
1 → Read/Write	Read (immediata);
0 → Read/Write	Write (immediata);
1 → Request	Request (immediata);

Microoperazioni speciali per AC → ALU e Risultato ALU → RBUS non sono strettamente necessarie, perche' le connessioni possono essere dirette

Macchine a Stati Finiti per CPU semplici

Trasformazione nel controllo dell'Unita' Operativa

Flusso dettagliato dei segnali di controllo



5 ingressi

Verificare che Reset e Wait siano sincronizzati

16 segnali di controllo UO

2 segnali di controllo memoria

Realizzazione Unita' di Controllo

Riassunto del capitolo

- ***Organizzazione fondamentale dei calcolatori di Von Neumann***

  Separazione tra processore e memoria

- ***Connessioni dell'Unita' Operativa***

- ***Organizzazione dell'Unita' di Controllo***

  Operazioni di trasferimento tra registri
  Microoperazioni