UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Il Processore

Luigi Palopoli

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Obbiettivi

- In questa serie di lezioni cercheremo di capire come e' strutturato un processore
- Le informazioni che vedremo si integrano con quanto visto sulle reti logiche
- Facciamo riferimento ad un insieme di istruzioni ridotto
  - ISTRUZIONI DI ACCESSO ALLA MEMORIA:
    - ✓ **lw, sw**
  - ISTRUZIONI MATEMATICHE e LOGICHE:
    - ✓ **add, sub, and, or, slt**
  - ISTRUZIONI DI SALTO
    - ✓ **beq, j**
- Le altre istruzioni si implementano con tecniche simili

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Panoramica Generale

- Nell'esecuzione delle istruzioni, per come e' progettato l'ISA, vi sono molti tratti comuni
- Le prime due fasi, *per ogni istruzione,* sono
  - Prelievo dell'istruzione dalla memoria
  - Lettura del valore di uno o piu' registri operandi che vengono estratti direttamente dai campi dell'istruzione
- I passi successivi dipendono dalla specifica istruzione ma, fortunatamente, sono molto simili per ciascuna delle tre classi indiividuate

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

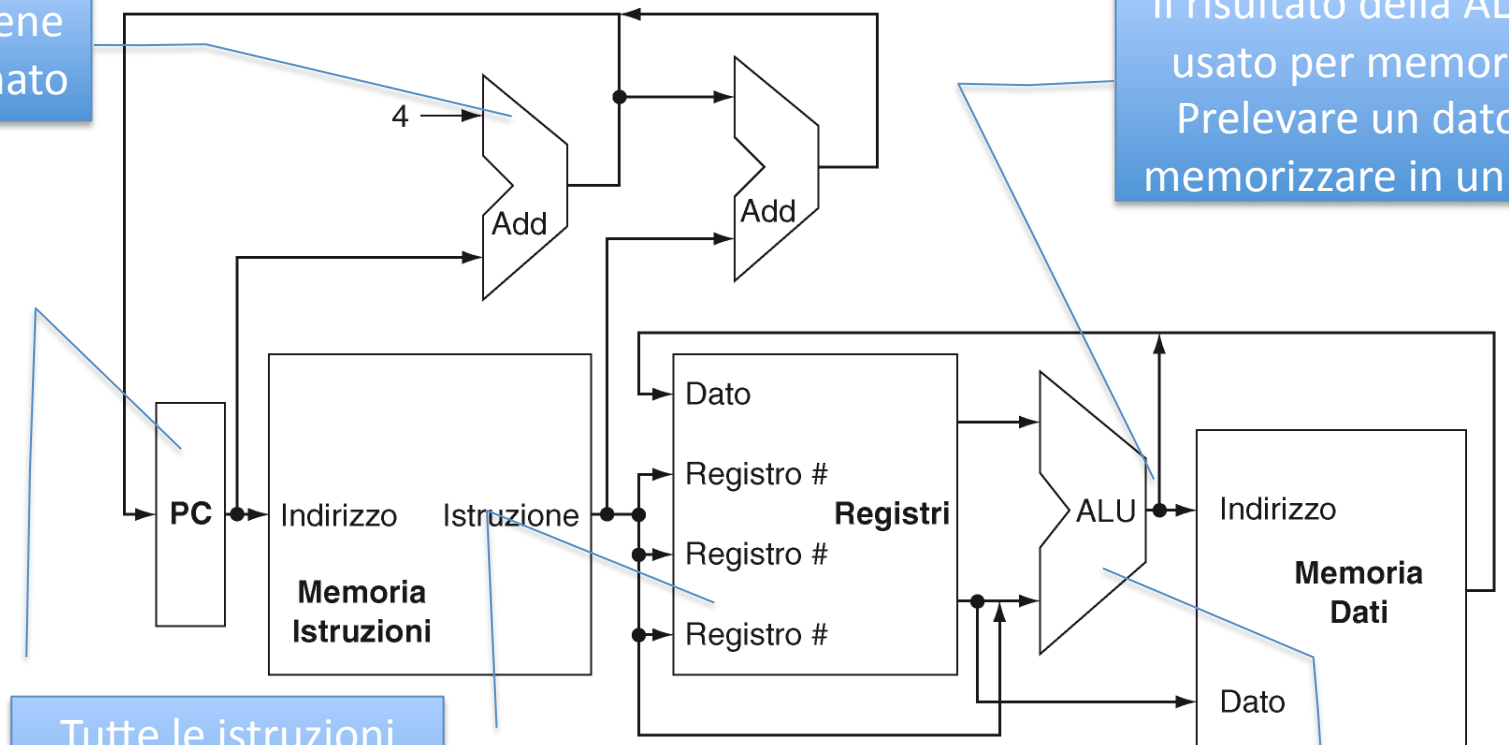
# Passi per ciascuna classe

- Tutti I tipi di istruzioni  tranne la jusano la ALU (unita' locico aritmetica) dopo aver letto gli operandi
  - Le istruzioni di accesso alla memoria per calcolare l'indirizzo
  - Le istruzioni aritmetico/lociche per eseguire quanto previsto dall'istruzione
  - I salti condizionati per effettuare il confronto
- Dopo l'uso della ALU il comportamento differisce per le tre classi
  - Le istruzioni di accesso alla memoria richiedono o salvano il dato in memoria
  - Le istruzioni artimetiche/logiche memorizzano il risultato nel registro target
  - Le istruzioni di salto condizionato cambiano il valore del registro PC secondo l'esito del confronto

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Schema di base

- Di seguito illustriamo la struttura di base della parte operativa (o datapath) per le varie istruzioni




Il PC viene aggiornato

Il risultato della ALU viene usato per memorizzare/ Prelevare un dato o per memorizzare in un registro

Tutte le istruzioni usano il PC per prelevare l'istruzione

La seconda fase prevede il caricamento degli operandi

III Fase: lavora la ALU

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Pezzi mancanti

- La figura precedente e' incompleta e crea l'impressione che ci sia un flusso continuo di dati
- In realta' ci sono punti in cui i dati arrivano da diverse sorgenti e bisogna sceglierne una (punto di decisione)
- E' il caso per esempio dell'incremento del PC
  - Nel caso "normale" il suo valore proviene da un circuito addizionatore (che lo fa puntare alla word successiva a quella appena letta)
  - Nel caso di salto il nuovo indirizzo viene calcolato a partire dall'offset contenuto nel campo dell'istruzione

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

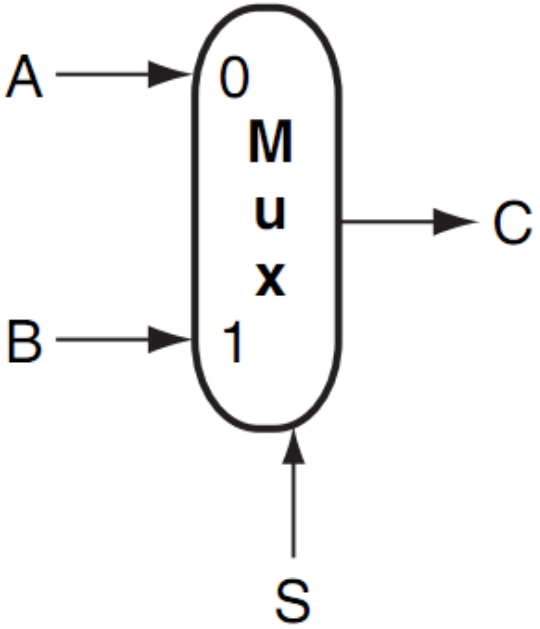
# Pezzi mancanti

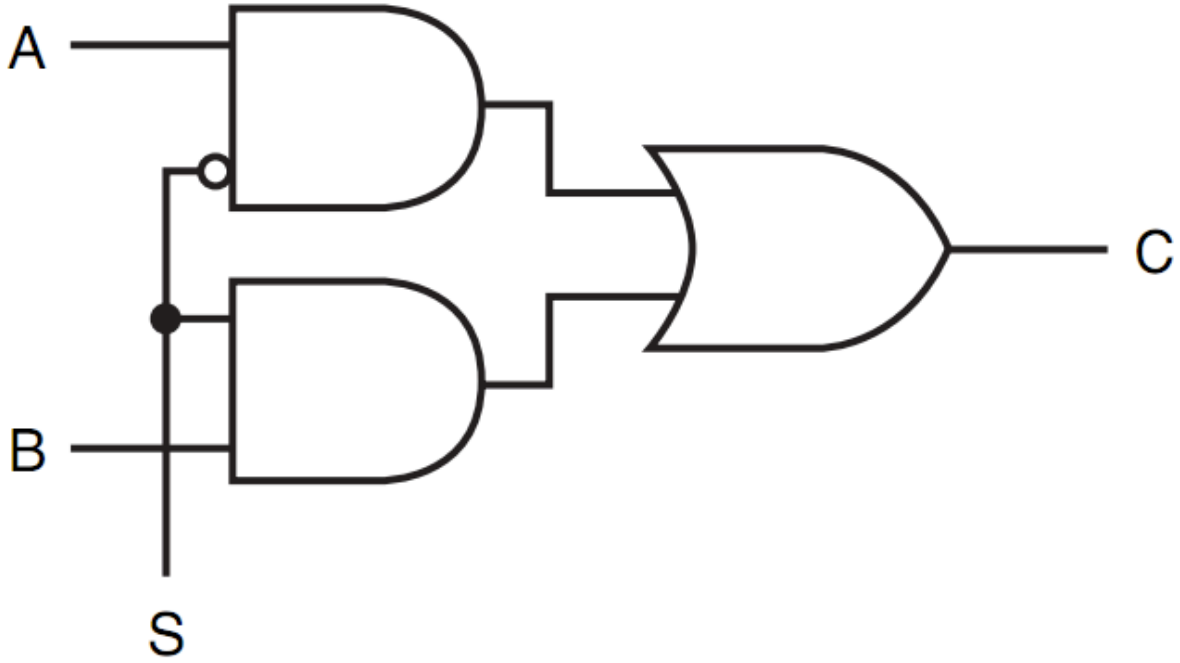
- Altro esempio
  - Il secondo operando della ALU puo' provenire dal banco registri (per istruzioni di tipo **R**) o dal codice dell'istruzione stessa (per istruzioni di tipo **I**)
- Per selezionare quale delle due opzioni scegliere viene impiegato un particolare rete combinatoria (multiplexer) che funge da selettore dei dati

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Il multiplexer

- Il multiplexer ha due (o piu') ingressi dati e un ingresso di controllo
- Sulla base dell'ingresso di controllo si decide quale degli input debba finire in output



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Ulteriori pezzi mancanti

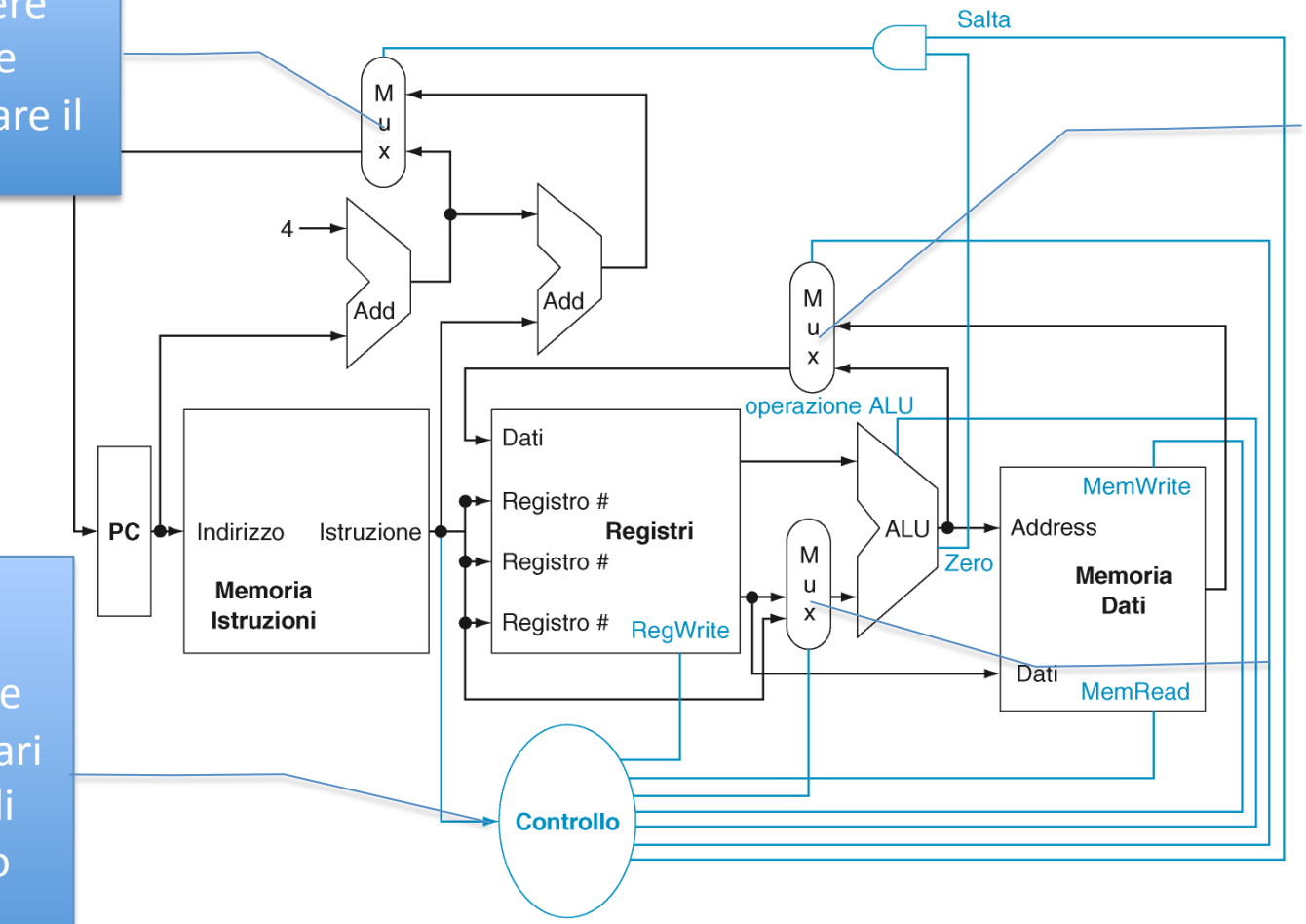
- Le linee di controllo dei multiplexer vengono impostate sulla base del tipo di istruzionie
- I vari blocchi funzionali hanno ulteriori ingressi di controllo
  - la ALU ha diversi ingressi per decidere quale operazione effettuare
  - Il banco registri ha degli ingressi per decidere se scrivere o meno inun registro
  - La memoria dati ha degli ingressi per decidere se vogliamo effettuare letture o scritture
- Per decidere come impiegare I vari ingressi di controllo abbiamo bisogno di un'unita' che funga da "direttore d'orchestra"

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Una figura piu' completa

Mux per decidere come aggiornare il PC

Nel registro target memorizziam oil risultato della ALU o quello che preleviamo dalla memoria?




Unita' che genera I vari segnali di controllo

Secondo operando registro o immediato?

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Informazioni di base

- ASSUNZIONE SEMPLIFICATIVA:
  - Il Processore lavora sincronizzandosi con i cicli di clock
  - Per il momento facciamo l'assunzione semplificativa che tutte le istruzioni si svolgano in un singolo ciclo di clock (lungo abbastanza)
- Prima di entrare nella descrizione dei vari componenti ci occorre richiamare alcuni concetti di reti logiche

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

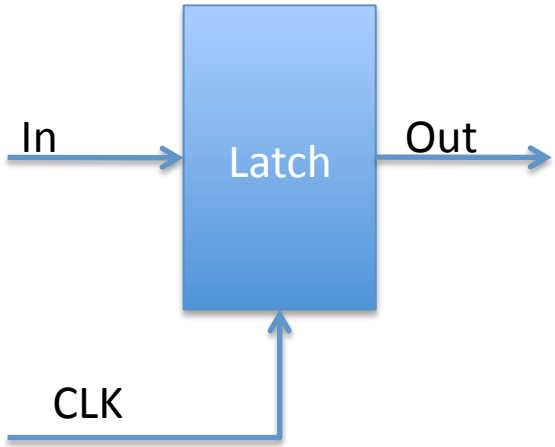
# Reti logiche

- Chiamiamo *rete logica combinatoria* un circuito composto di porte logiche che produce un output funzione dell'input
  - Esempio il multiplexer visto prima
- Vi sono inoltre elementi che chaimiamo di stato
  - In sostanza se a un certo punto salviamo il valore degli elementi di stato e poi lo ricarichiamo il computer riparte esattamente da dove si era interrotto
  - Nel nostro caso elementi di stato sono registri, memoria dati, memoria istruzioni
- Gli elementi di stato sono detti *sequenziali* perche' l'uscita a un inggresso dipende dalla storia (sequenza) degli ingressi
- Gli elementi di stato hanno due ingressi almeno:
  - il valore da immettere nello stato
  - Un clock a cui sincronizzare le transizioni di stato

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Flip-Flop

- L'elemento base per memorizzare un bit e' un circuito sequenziale chiamato *flip-flop d-latch*




- I registri possono essere ottenuti come array di latch (o in altri modi simili)
- Terminologia
  - Asserito: segnale logico a livello alto
  - Non Asserito: segnale logico a livello basso

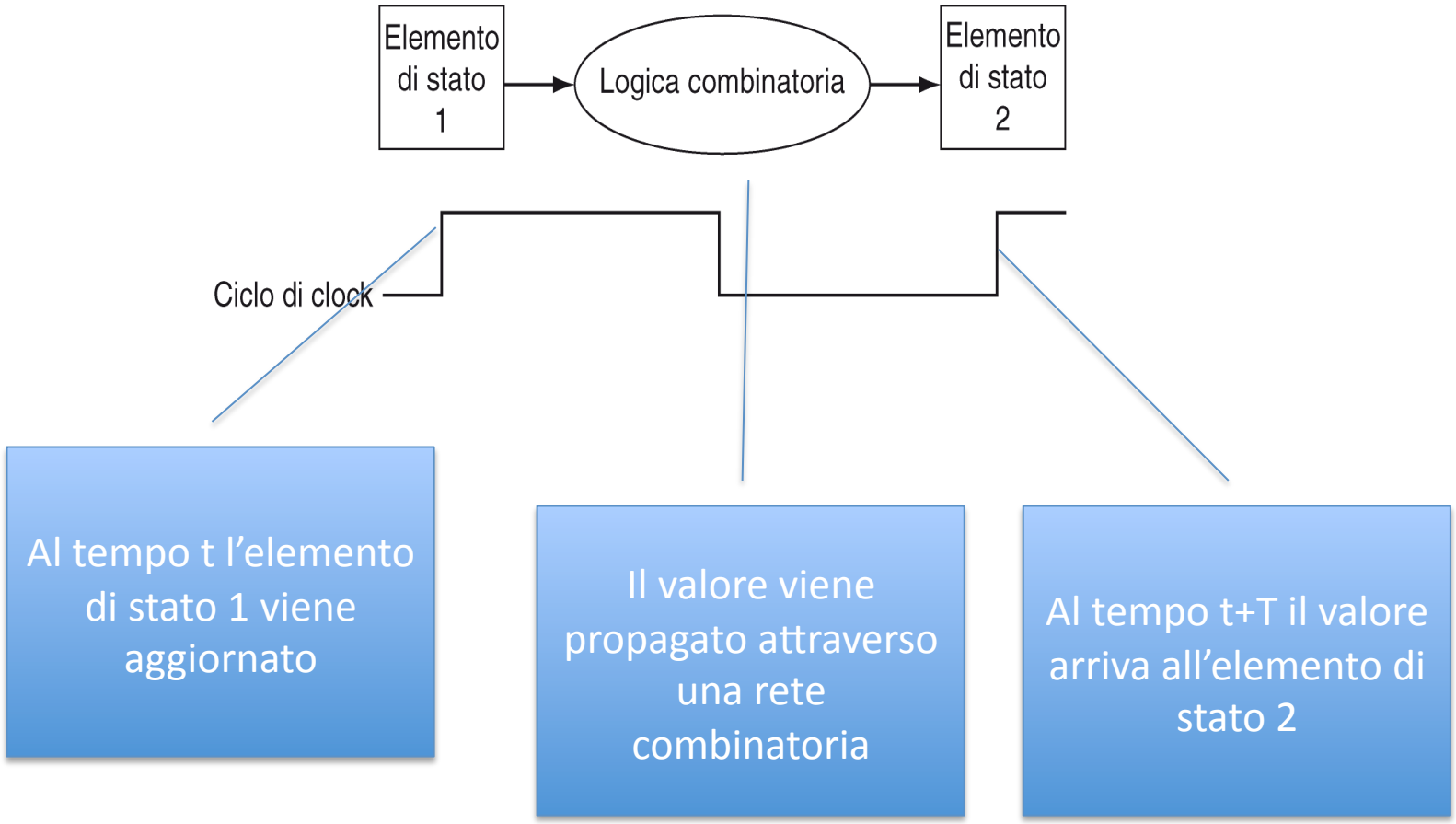
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Temporizzazione

- La metodologia di temporizzazione ci dice quando i segnali possono essere letti o scritti in relazione al clock
- E' importante stabilire una temporizzazione
  - Se leggo e scrivo su registro, devo sapere se il dato che leggo e' quello precedente o successivo alla scrittura
- La tecnica di temporizzazione piu' usata e' quella sensibile ai fronti
  - Il dato viene memorizzato in corrispondenza della salita o della discesa del fronte di clock
- I dati presi da elementi di stato sono relativi al ciclo precedente

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Esempio



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

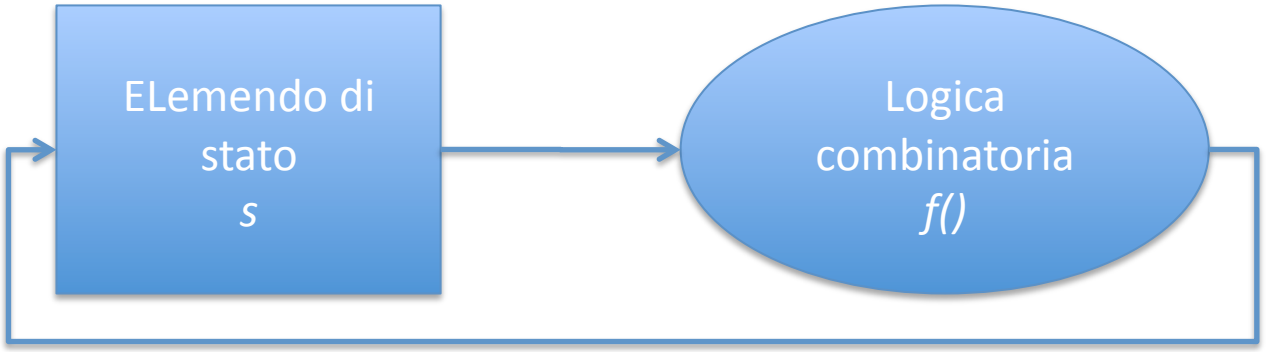
# Considerazioni

- Il tempo di clock T deve essere scelto in modo da dare tempo ai dati di attraversare la rete combinatoria
- Nel caso del MIPS 32 gli elementi di stato contengono tipicamente 32 bit
- La metodologia di memorizzazione sensibile ai clock permette di realizzare interconnessioni che, a prima vista, creerebbero dei loop di retroazione che renderebbero imprdicibile l'evoluzione del sustema

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Esempio

- Consideriamo un caso come questo




- Se non avessimo temporizzazioni precise dovremmo scrivere

$$s = f(s)$$

- Grazie alla temporizzazione sensibile al clock abbiamo

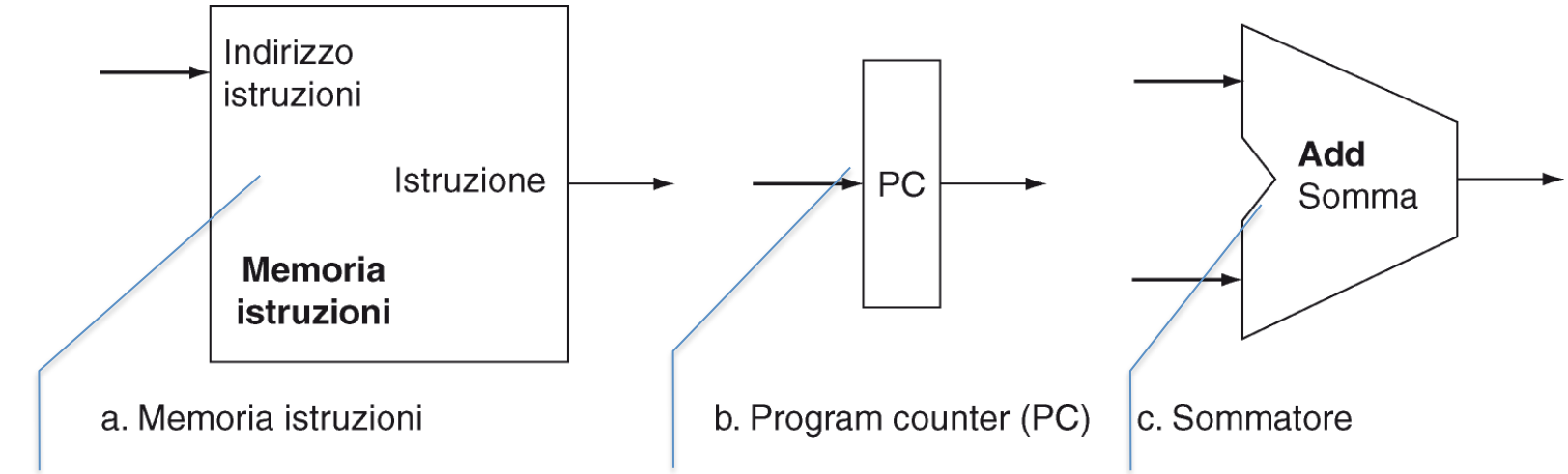
$$s(t+T) = f(s(t))$$

che mi fa pensare a evoluzione ben determinata

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Realizzazione del datapath

- Passiamo ora in rassegna le varie componenti che ci servono per la realizzazione del datapath




Memoria dove sono memorizzate le istruzioni

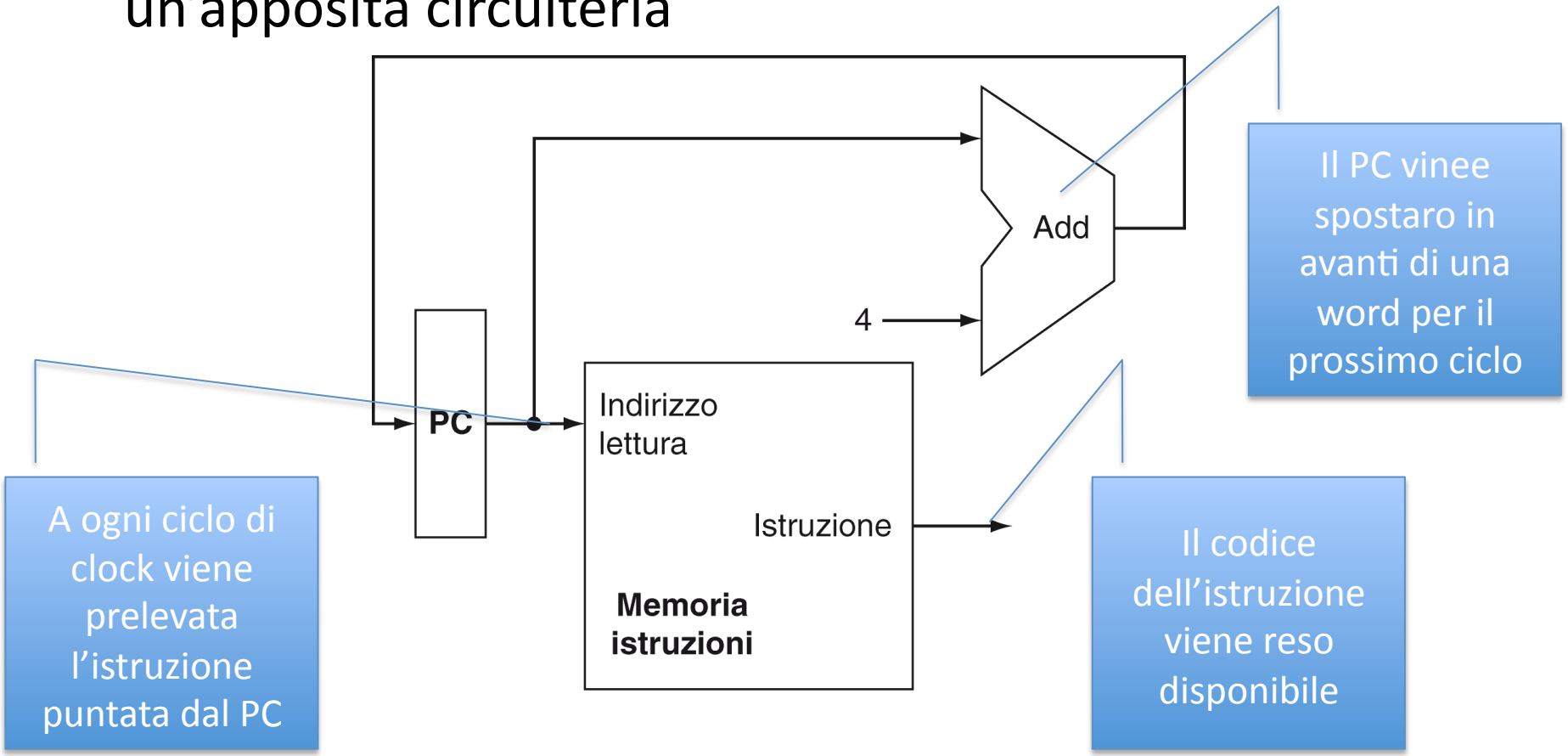
Registro che contiene l'indirizzo dell'istruzione da caricare

Addizionatore (ALU specializzata) per incrementare il PC

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Prelievo dell'istruzione

- Usando gli elementi che abbiamo visto possiamo mostrare come effettuare il prelievo dell'istruzione con un'apposita circuiteria



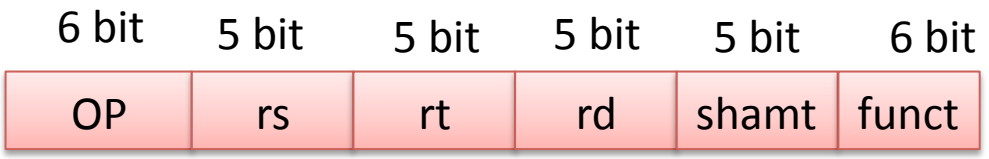
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Istruzioni di tipo R

- Cominciamo dal vedere come vengono eseguite le istruzioni di tipo R
- Si tratta di istruzioni aritmetiche/logiche che operano tra registri e producono un risultato che viene memorizzato in un registro
- Es

```
add $t0, $s1, $s2
```

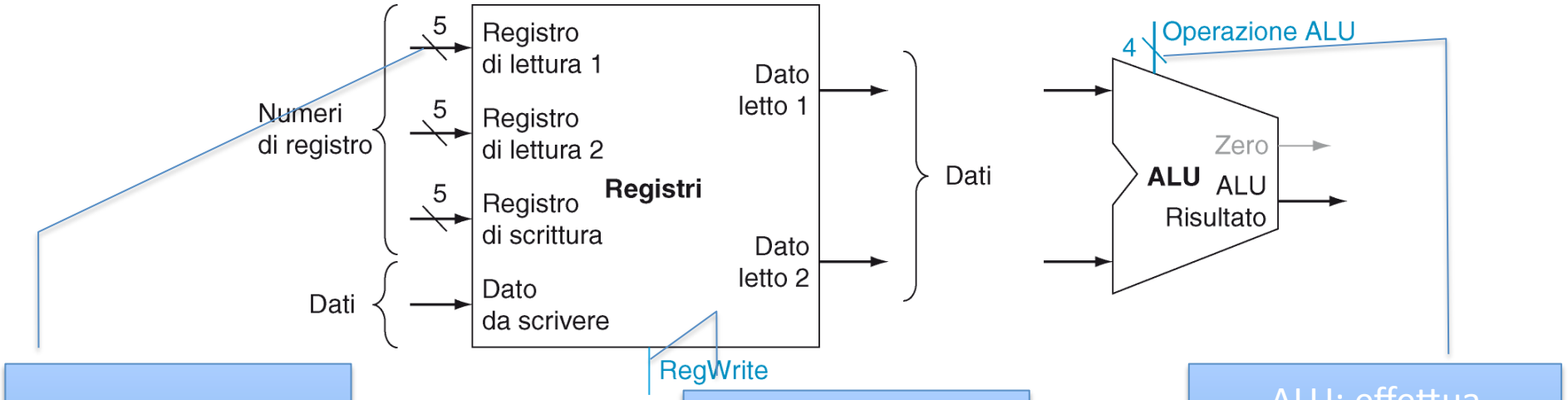
- Il codice ha la forma



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Blocchi funzionali richiesti

- Per effettuare questi calcoli ho bisogno di due ulteriori blocchi funzionali.




Banco registri: da in output I registri specificati nel registro di lettura q e 2

tri

Se abilitato in scrittura scrive nel regitro specificato il dato in ingresso

b. ALU

ALU: effettua l'operazione arimetica codificata in 4 bit. Setta un bit in uscita se il risultato e' zero

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Istruzioni load/store

- Consideriamo ora anche le istruzioni load (lw) e store (sw)
- Forma generale

```
lw $t0, offset($t2)
sw $t0, offset($t2)
```

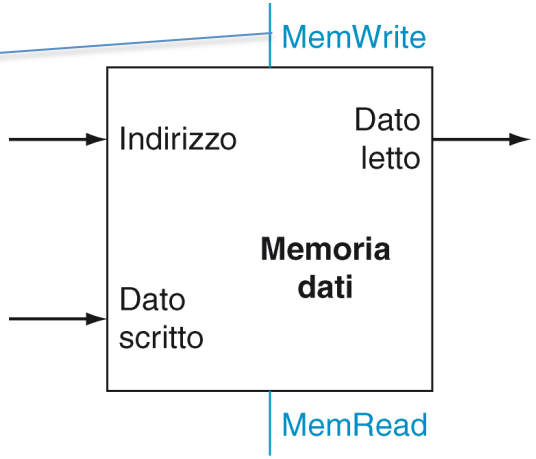

- Per entrambe si deve calcolare un indirizzo di memoria dato dalla somma di t2 con l'offset
- Nel secondo caso occorre leggere t0 dal register file
- Quindi per eseguire queste istruzioni ci serve ancora la ALU e il register file

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

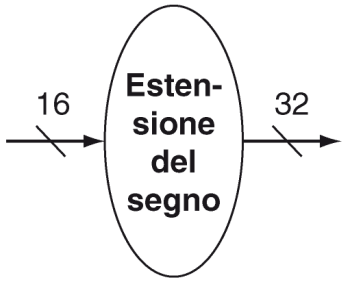
# Istruzioni load/store

- Notare che l'offset viene memorizzato in un campo a 16 bit che occorrera' estendere a 32 bit (replicando per 16 volte il bit di segno)
- In aggiunta alle componenti che abbiamo visto prima ci occorre un'unita' di memoria dati dove memorizzare eventualmente con *sw*

A differenza della memoria istruzioni quesa puo' essere usata in lettura e scrittura. Quindi ho bisogno di comandi appositi




a. Unità di memoria dati




b. Unità di estensione del segno

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Salto condizionato

- L'istruzione di salto condizionato ha la forma

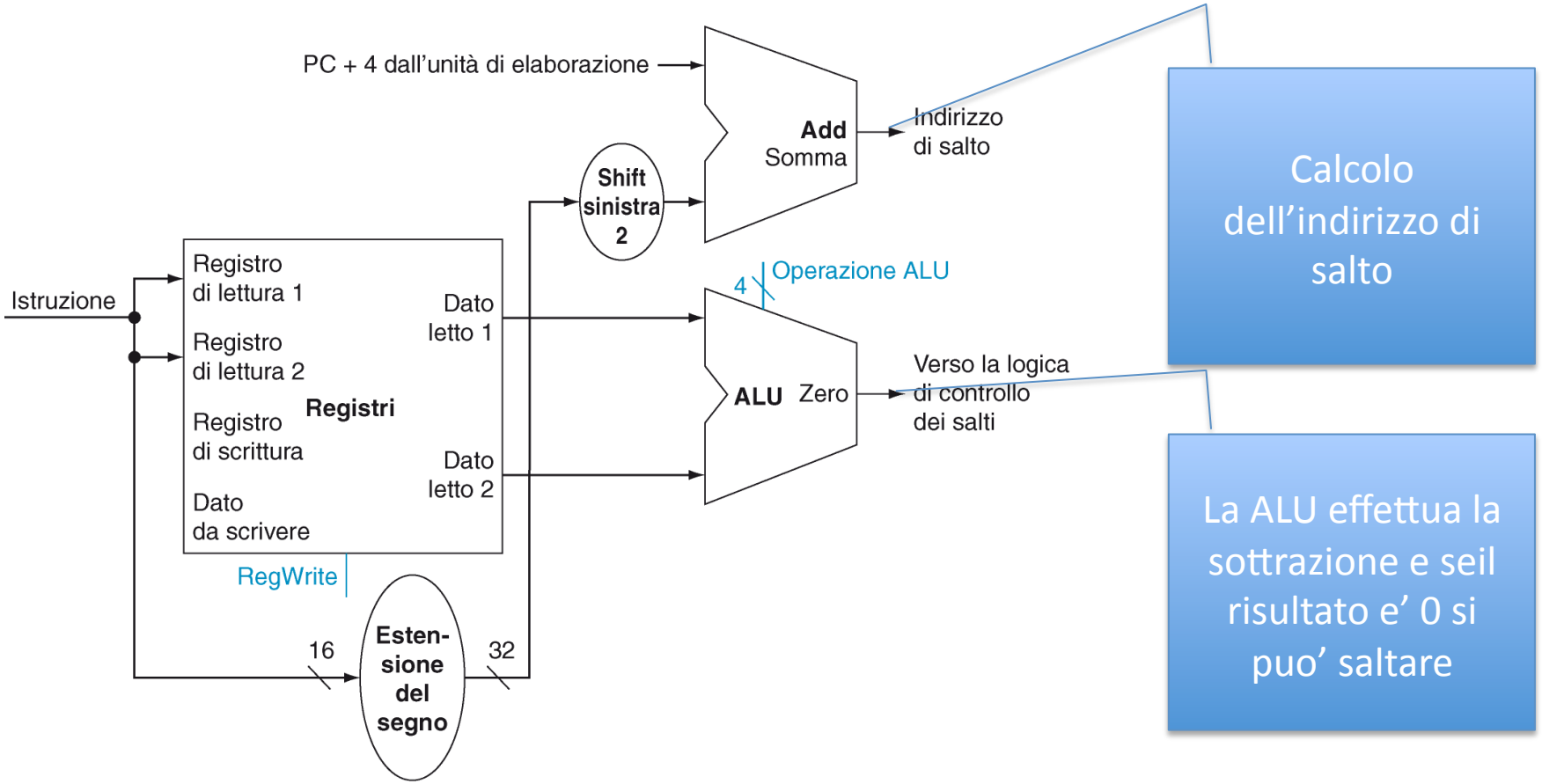
```
beq $t1, $t2, offset
```

  - Anche in questo caso bisogna sommare l'offset (dopo averlo esteso a 32 bit) all'attuale PC
  - Due attenzioni
    - Siccome normalmente il PC viene automaticamente aumentato di 4 tanto vale l'offset riferirlo gia' al PC aumentato
    - L'architettura automaticamente trasla l'offset di due bit (in modo da esprimerlo in word) e da estendere il range di indirizzi che si puoi raggiungere (rispetto all'espressione degli offset in byte)

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Salto condizionato

- Nell'esecuzione della *beq* occorre anche un meccanismo in base al quale decidere se aggiornare il PC a PC + 4 o a PC + 4 + offset



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

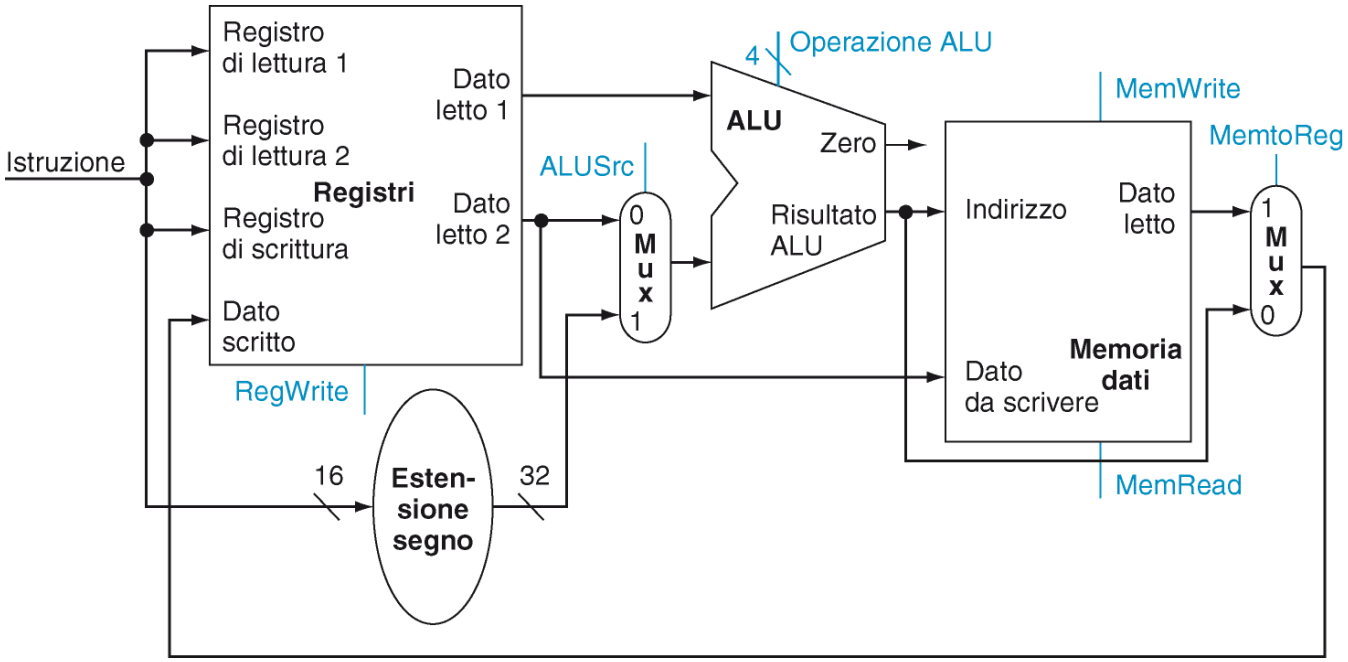
# Progetto di un'unita' di elaborazione

- Per il salto non condizionato basta sostiutire il campo offset (shiftato di due posizioni) al PC
- Siccome abbiamo il requisito di eseguire ogni istruzioni in un ciclo di clock, non possiamo usare un'unita' funzionale piu' di una volta in ogni ciclo
  - percio' dobbiamo distinguere memoria dati e memoria istruzioni
- Inoltre occorre condividere il piu' possibile le varie unita'

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Esempio

- Con questo circuito riusciamo a eseguire istruzioni di tipo R e istruzioni di trasferimento nella memoria




Per istruzione di tipo R:

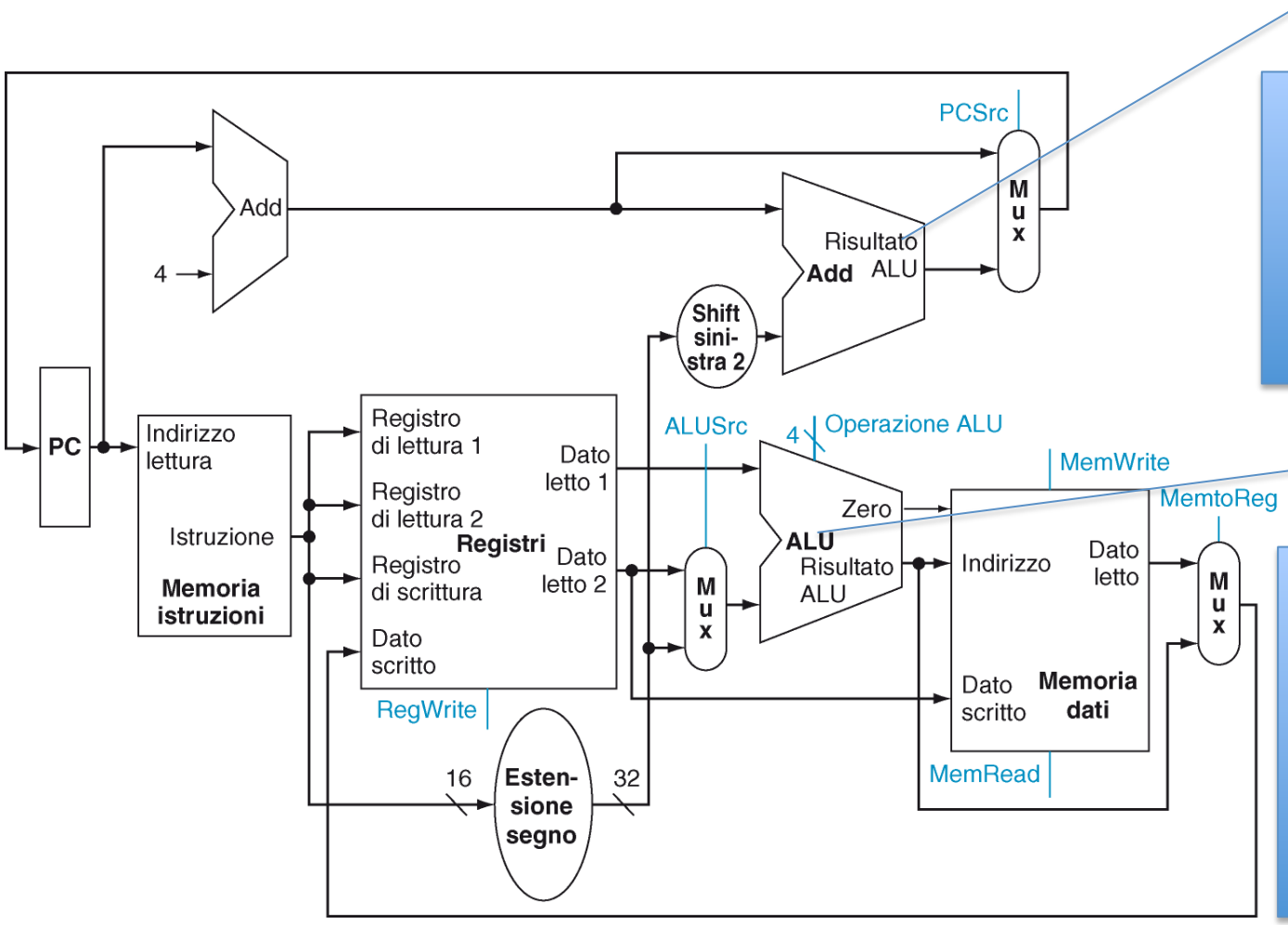
- ALUSrc=0
- REGwrite = 1
- MemtoReg=0

Per istruzione lw

- ALUSrc=1
- REGwrite = 1
- MemtoReg=0

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Un esempio piu' completo




Circuiteria per I salti condizionati

Con questo DATAPATH possiamo fare istruzioni di tipo R, memorizzazioni e salti condizionati

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Prima implementazione completa

- Per arrivare a una prima implementazione completa partiamo dal datapath mostrato e aggiungiamo la parte di controllo
- Implementeremo le istruzioni
  - ✓add, sub, and, or and slt
  - ✓lw, sw
  - ✓beq
- Successivamente vestremo come introdurre
  - ✓j

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Cominciamo dalla ALU

- La ALU viene impiegata per:
  - Effettuare operazioni logico-aritmetiche (tipo R), compreso slt
  - Calcolare indirizzi di memoria (per sw e lw)
  - Sottrazione per beq
- Per queste diverse operazioni abbiamo una diversa configurazione degli input di controllo (Linea controllo ALU)

| Linea controllo ALU | Operazione |
|---|---|
| 0000 | AND |
| 0001 | OR |
| 0010 | Somma |
| 0110 | Sottrazione |
| 0111 | SLT |
| 1100 | NOR |

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Ancora sul controllo della ALU

- Per generare i 4 bitdi controllo della ALU useremo una piccola unita' di controllo che riceve in ingresso
  - il campo funct prelevato dall'istruzione
  - due bit deti ALUop
    - ✓ALUop = 00 -> somma per istruzioni di sw e lw
    - ✓ALUOp = 01 -> sottrazione per beq
    - ✓ALUOp = 10 -> operazione di tipo R (specificato dal funct)

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Ancora sul controllo della ALU

## Tabella riassuntiva

| Codice operativo istruzione | ALUOp | Operazione eseguita dall'istruzione | Campo Funct | Operazione dell'ALU | Ingresso di controllo alla ALU |
|---|---|---|---|---|---|
| Lw | 00 | load di 1 parola | XXXXXX | somma | 0010 |
| Sw | 00 | store di 1 parola | XXXXXX | somma | 0010 |
| Branch on equal | 01 | salto condizionato all'uguaglianza | XXXXXX | sottrazione | 0110 |
| Tipo R | 10 | somma | 100000 | somma | 0010 |
| Tipo R | 10 | sottrazione | 100010 | sottrazione | 0110 |
| Tipo R | 10 | AND | 100100 | AND | 0000 |
| Tipo R | 10 | OR | 100101 | OR | 0001 |
| Tipo R | 10 | set on less than | 101010 | set on less than | 0111 |

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Decodifica a livelli multipli

- Quello che aviamo visto e' un sistema di decodifica e generazione dei comandi a due livelli
  - **Livello 1**: genera i segnali di controllo ALUOp per l'unita' di controllo della ALU
  - **Livello 2:** (unita' di controllo ALU) genera I segnali di controllo per la ALU

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Unita' di controllo dell'ALU

- I segnali di controllo della ALU sono generati da una rete logica combinatoria (unita' di controllo della ALU)
- Bisognerebbe elencare tutte le combinazioni di ingresso di ALUop e dei campi funct (8 bit in tutto)
- Per evitare di elencare tutte le combinazioni (che sono 256) useremo X come wildcard (un po' come * nel filesystem).

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Unita' di controllo dell'ALU

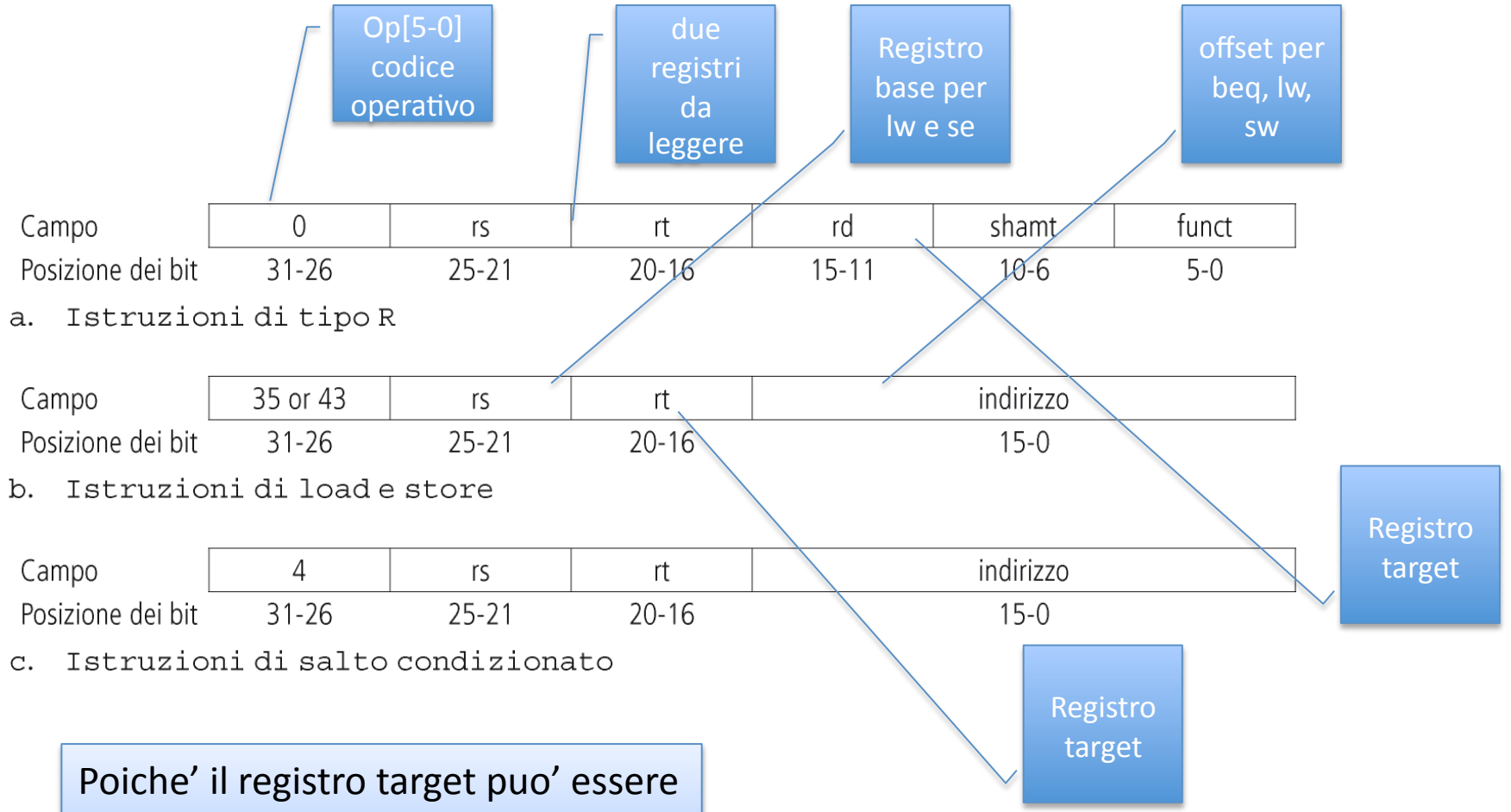
Tabella di verita' (compressa):

| ALUOp | | Campo Funct | | | | | | Operazione |
|---|---|---|---|---|---|---|---|---|
| ALUOp1 | ALUOp2 | F5 | F4 | F3 | F2 | F1 | F0 | |
| 0 | 0 | X | X | X | X | X | X | 0010 |
| 0 | 1 | X | X | X | X | X | X | 0110 |
| 1 | 0 | X | X | 0 | 0 | 0 | 0 | 0010 |
| 1 | X | X | X | 0 | 0 | 1 | 0 | 0110 |
| 1 | 0 | X | X | 0 | 1 | 0 | 0 | 0000 |
| 1 | 0 | X | X | 0 | 1 | 0 | 1 | 0001 |
| 1 | X | X | X | 1 | 0 | 1 | 0 | 0111 |

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Unita' di controllo principale

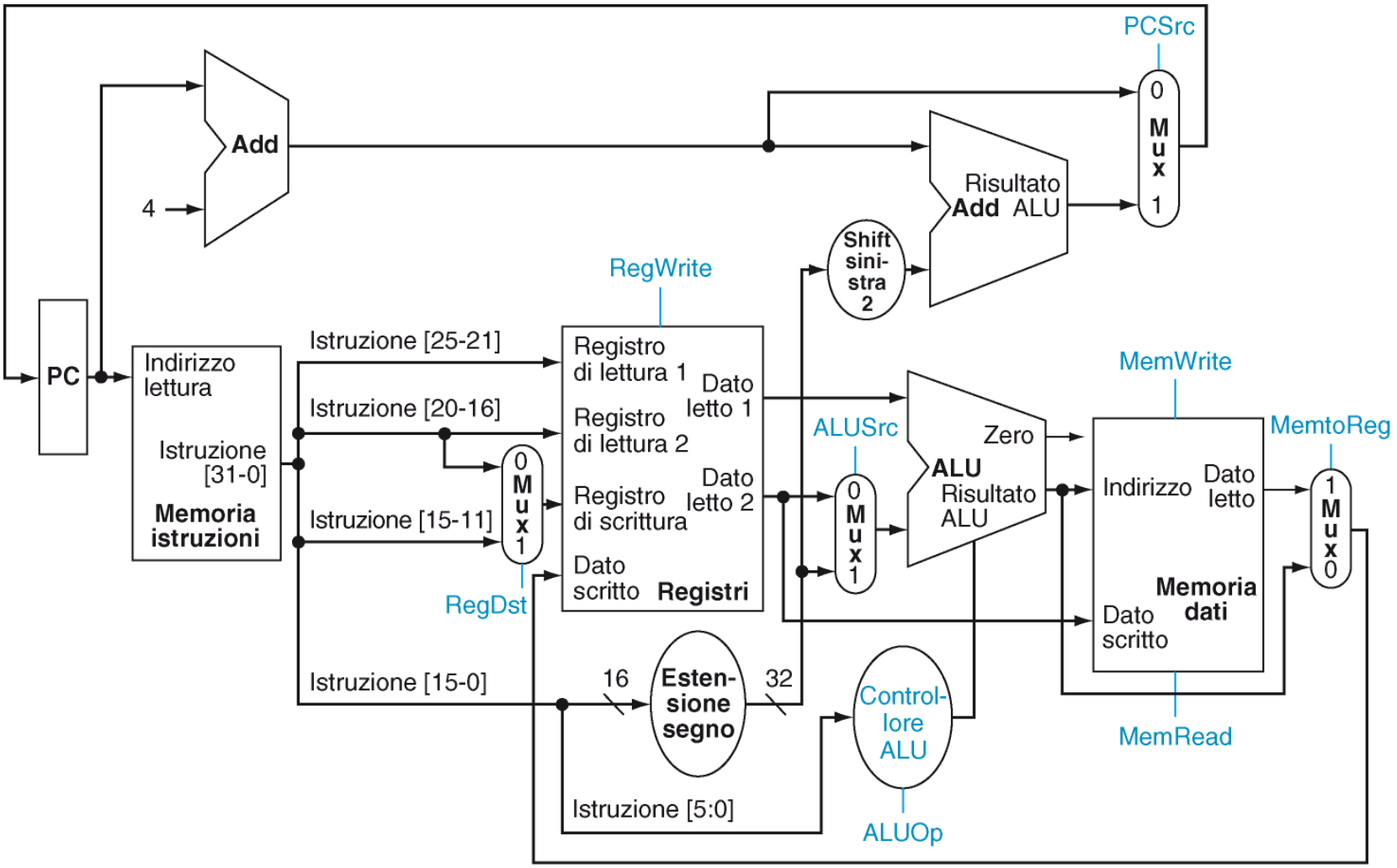
- Riguardiamo i campi.




Poiche' il registro target puo' essere in posizione diversa occorre un Mux

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Panoramica dei segnali di controllo



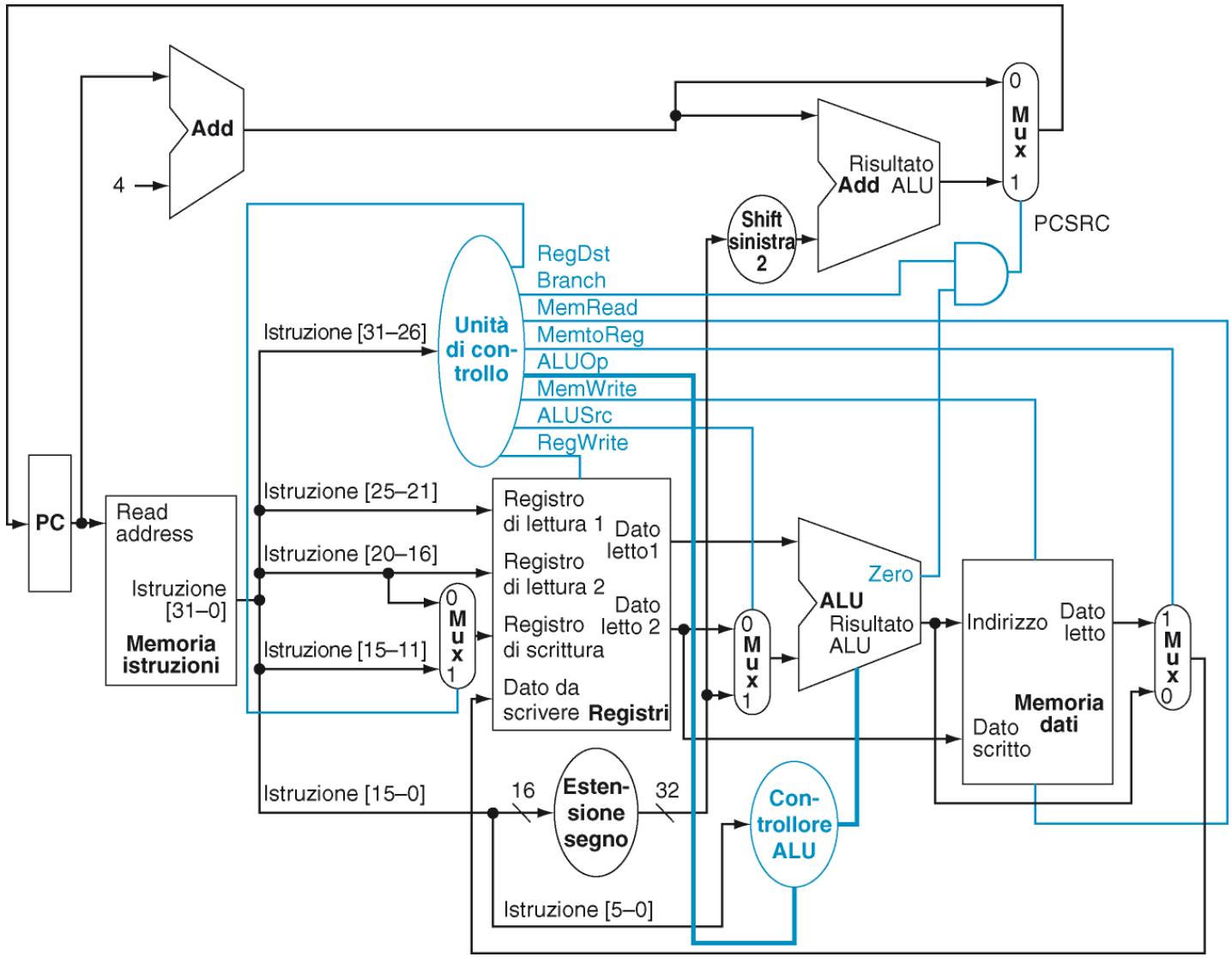
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Tabella riassuntivo dei segnali di controllo

| Nome del segnale | Effetto quando non asserito | Effetto quando asserito |
|---|---|---|
| RegDst | Il numero del registro di scrittura proviene dal campo rt (bit 20-16) | Il numero del registro di scrittura proviene dal campo rd (bit 15-11) |
| RegWrite | Nulla | Il dato viene scritto nel register file nel registro individuato dal numero del registro di scrittura |
| ALUSrc | Il secondo operando della ALU proviene dalla seconda uscita del register file (Dato letto 2) | Il secondo operando della ALU proviene dall'estensione del segno dei 16 bit meno significativi dell'istruzione |
| PCSrc | Nel PC viene scritta l'uscita del sommatore che calcola il valore di PC + 4 | Nel PC viene scritta l'uscita del sommatore che calcola l'indirizzo di salto |
| MemRead | Nulla | Il dato della memoria nella posizione puntata dall'indirizzo viene inviato in uscita sulla linea «dato letto» |
| MemWrite | Nulla | Il contenuto della memoria nella posizione puntata dall'indirizzo viene sostituito con il dato presente sulla linea «dato scritto» |
| MemtoReg | Il dato inviato al register file per la scrittura proviene dalla ALU | Il dato inviato al register file per la scrittura proviene dalla Memoria Dati |

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Schema complessivo



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# L'unita' di controllo

- L'unita' di controllo genera tutti I segnali di controllo
- Ancora e' una rete combinatoria che prende come input il codice operativo dell'istruzione e genera I comandi del caso, secondo la seguente tabella

| Istruzione | RegDst | ALUSrc | MemtoReg | RegWrite | MemRead | MemWrite | Branch | ALUOp1 | ALUOp0 |
|---|---|---|---|---|---|---|---|---|---|
| Tipo R | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| `lw` | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| `sw` | X | 1 | X | 0 | 0 | 1 | 0 | 0 | 0 |
| `beq` | X | 0 | X | 0 | 0 | 0 | 1 | 0 | 1 |

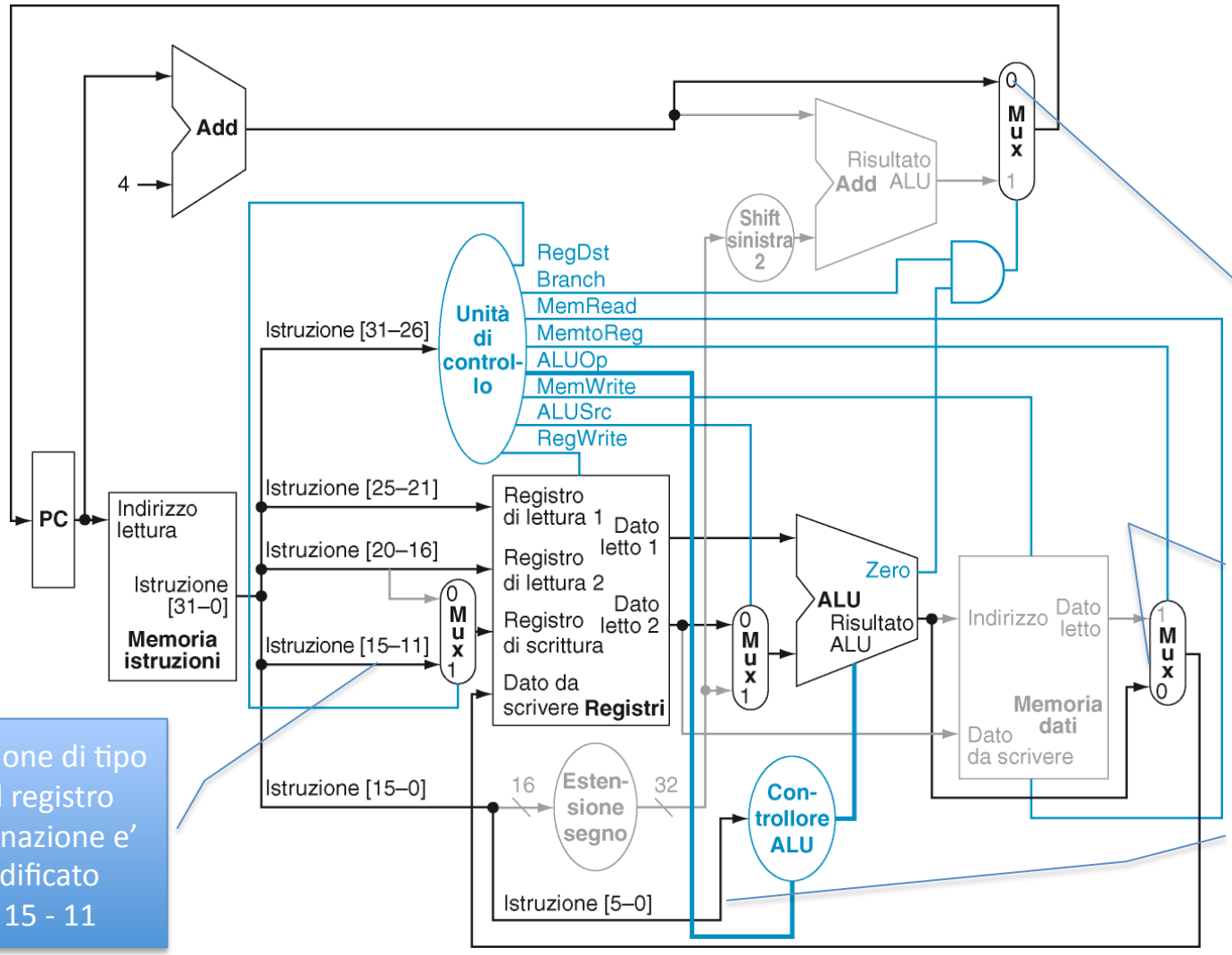
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Partiamo dalla ADD

- Per eseguire un'istruzione di tipo R (come *add $t1,$t2, $t3*) occorre
  1. prelevare l'istruzione dalla memoria e incrementare il PC di 4
  2. leggere t2 e t3 dal register file
  3. attivare la ALU con in input i dati dal register file
  4. memorizzare il risultato nel registro destinazione
- Tutto questo avviene in un solo ciclo

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Passi sul processore




Prelievo: nel Mux rimane attivoil ramo alto

In questo mux e' attiva la linea bassa (non uso la memoria per il risultato)

Istruzione di tipo R: il registro dedtinazione e' codificato in 15 - 11

Istruzione di tipo R: il controllore della ALU usa i bit funct

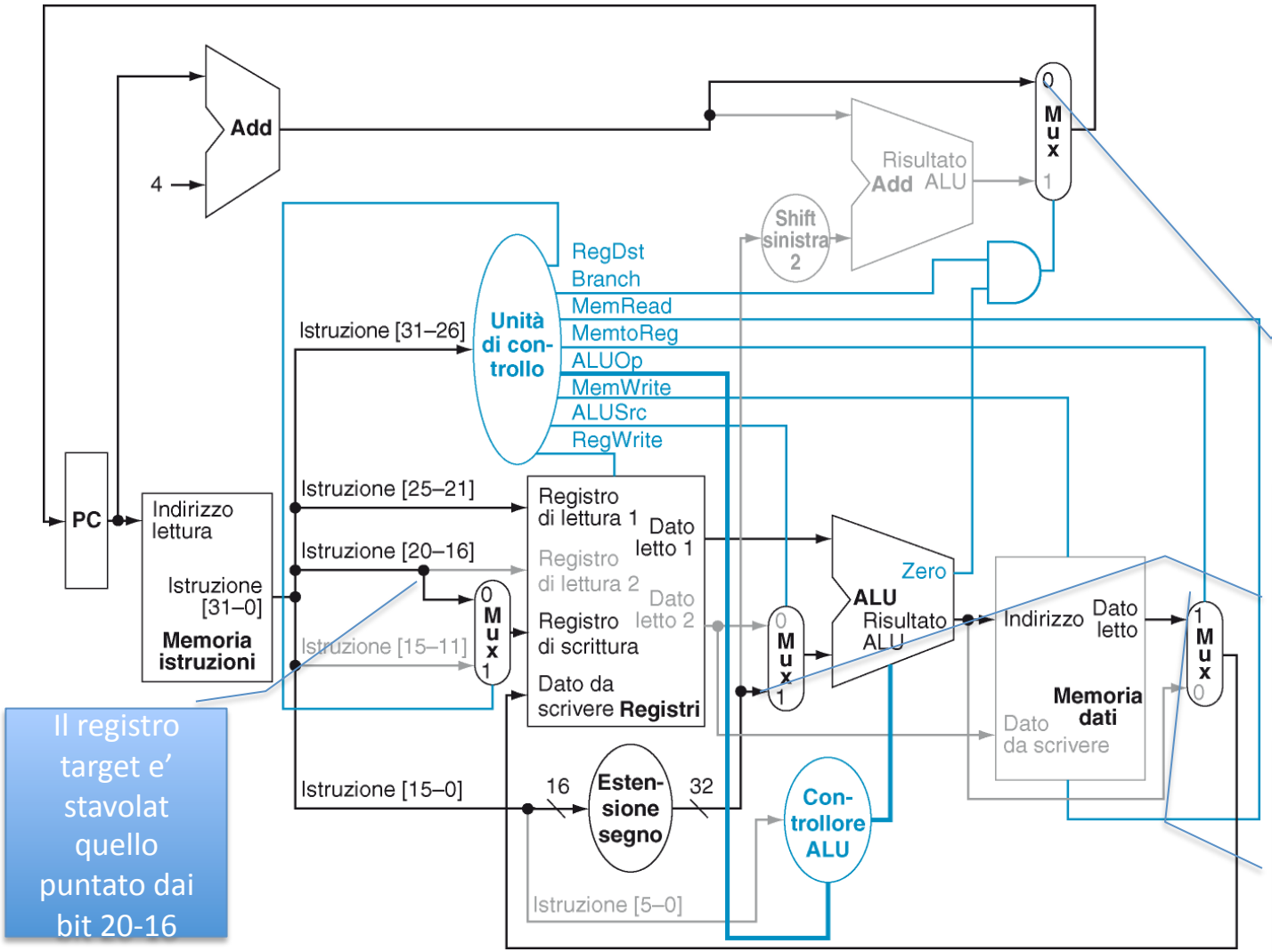
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Altro esempio

- Consideriamo ora l'istruzione
  lw $t1, offset($t2)
- Fasi
  1. la fase di prelievo e' uguale a prima
  2. prelevare t2 dal register file
  3. sommare  t2 ai campi offset dell'istruzione (I 16 bit meno significativi)
  4. l'indirizzo viene usato per memoria dati
  5. il dato prelevato dalla memoria dati e memorizzato in t2

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Passi sul processore




Prelievo: nel Mux rimane attivoil ramo alto

In questo mux e' attiva la linea 1 (secondo input dato da offset)

In questo mux, stavolta, il dato che passa e' quello sopra (provenienza memoria)

Il registro target e' stavolat quello puntato dai bit 20-16

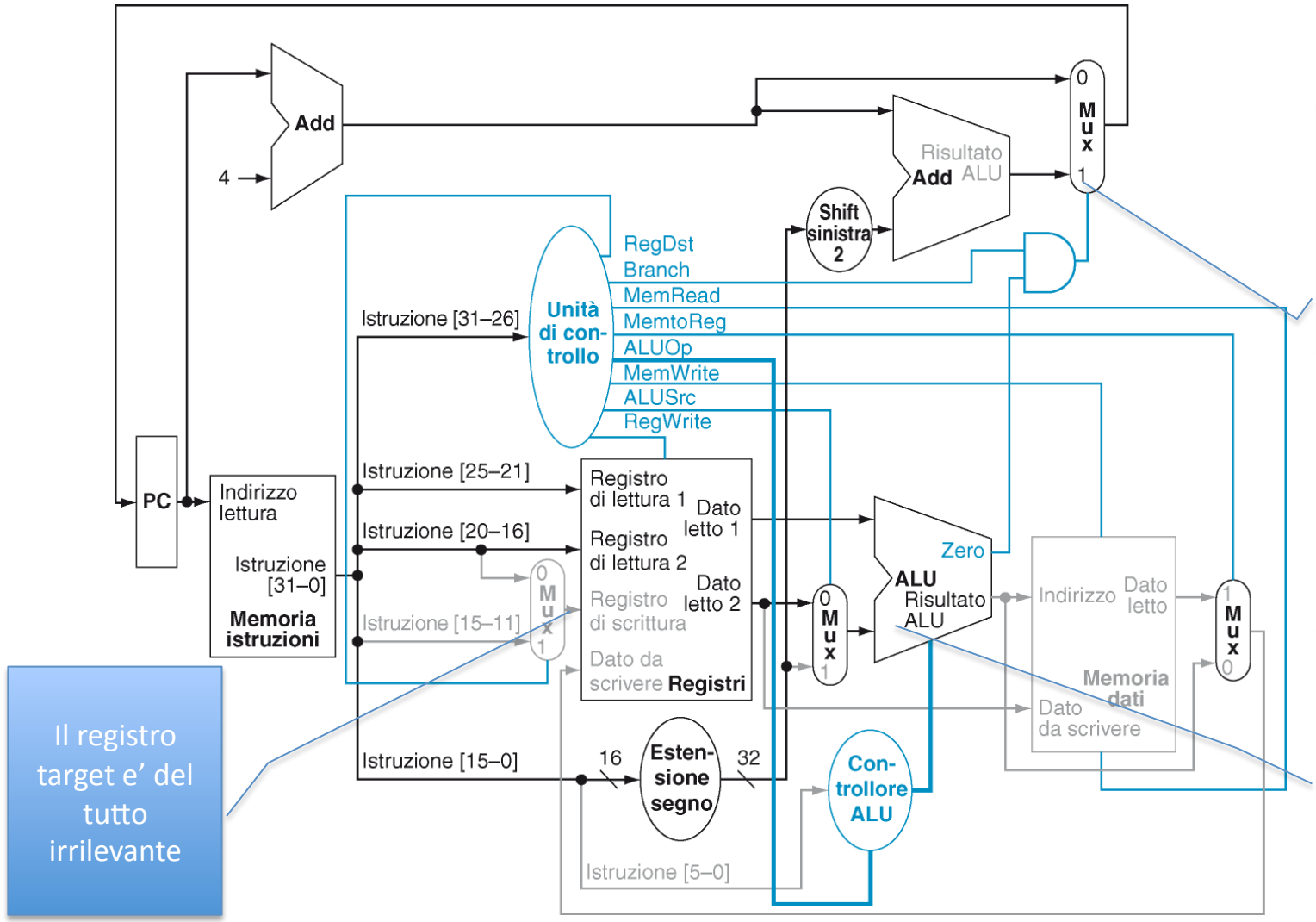
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# E ora il salto condizionale

- Consideriamo ora l'istruzione
  *beq $t1, $t2, offset*
- Fasi
  1. la fase di prelievo e' uguale a prima
  2. prelevare t1 e t2 dal register file
  3. la ALU sottrare t1 da t2. Il valore di PC+4 viene sommato all'offset (esteso a 32 bit) e shiftato di due volte (per indirizzare word e non in byte)
  4. la linea 0 della ALU viene usata per decidere a cosa settare il PC

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Sul processore




Con branch settato se il risultato dela ALU e' zero si attiva il ramo uno

Il registro target e' del tutto irrilevante

La ALU sottrae I due registri

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Implementazione

- Dopo aver capito a cosa servono le I vari comandi possiamo passare all'implementazione secondo la tabella gia' vista in precedenza

| Istruzione | RegDst | ALUSrc | MemtoReg | RegWrite | MemRead | MemWrite | Branch | ALUOp1 | ALUOp0 |
|---|---|---|---|---|---|---|---|---|---|
| Tipo R | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| `lw` | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| `sw` | X | 1 | X | 0 | 0 | 1 | 0 | 0 | 0 |
| `beq` | X | 0 | X | 0 | 0 | 0 | 1 | 0 | 1 |

- La prima colonna di questa tabella si traduce nel codice operativo (bit da 0 a 5 dell'istruzione)

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Implementazione

- A questo punto gli input e gli output sono quelli specificati nella seguente tabella:

| Input o Output | Nome del segnale | Formato R | `lw` | `sw` | `beq` |
|---|---|---|---|---|---|
| Input | Op5 | 0 | 1 | 1 | 0 |
| | Op4 | 0 | 0 | 0 | 0 |
| | Op3 | 0 | 0 | 1 | 0 |
| | Op2 | 0 | 0 | 0 | 1 |
| | Op1 | 0 | 1 | 1 | 0 |
| | Op0 | 0 | 1 | 1 | 0 |
| Output | RegDst | 1 | 0 | X | X |
| | ALUSrc | 0 | 1 | 1 | 0 |
| | MemtoReg | 0 | 1 | X | X |
| | RegWrite | 1 | 1 | 0 | 0 |
| | MemRead | 0 | 1 | 0 | 0 |
| | MemWrite | 0 | 0 | 1 | 0 |
| | Branch | 0 | 0 | 0 | 1 |
| | ALUOp1 | 1 | 0 | 0 | 0 |
| | ALUOp2 | 0 | 0 | 0 | 1 |

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Salto incondizionato

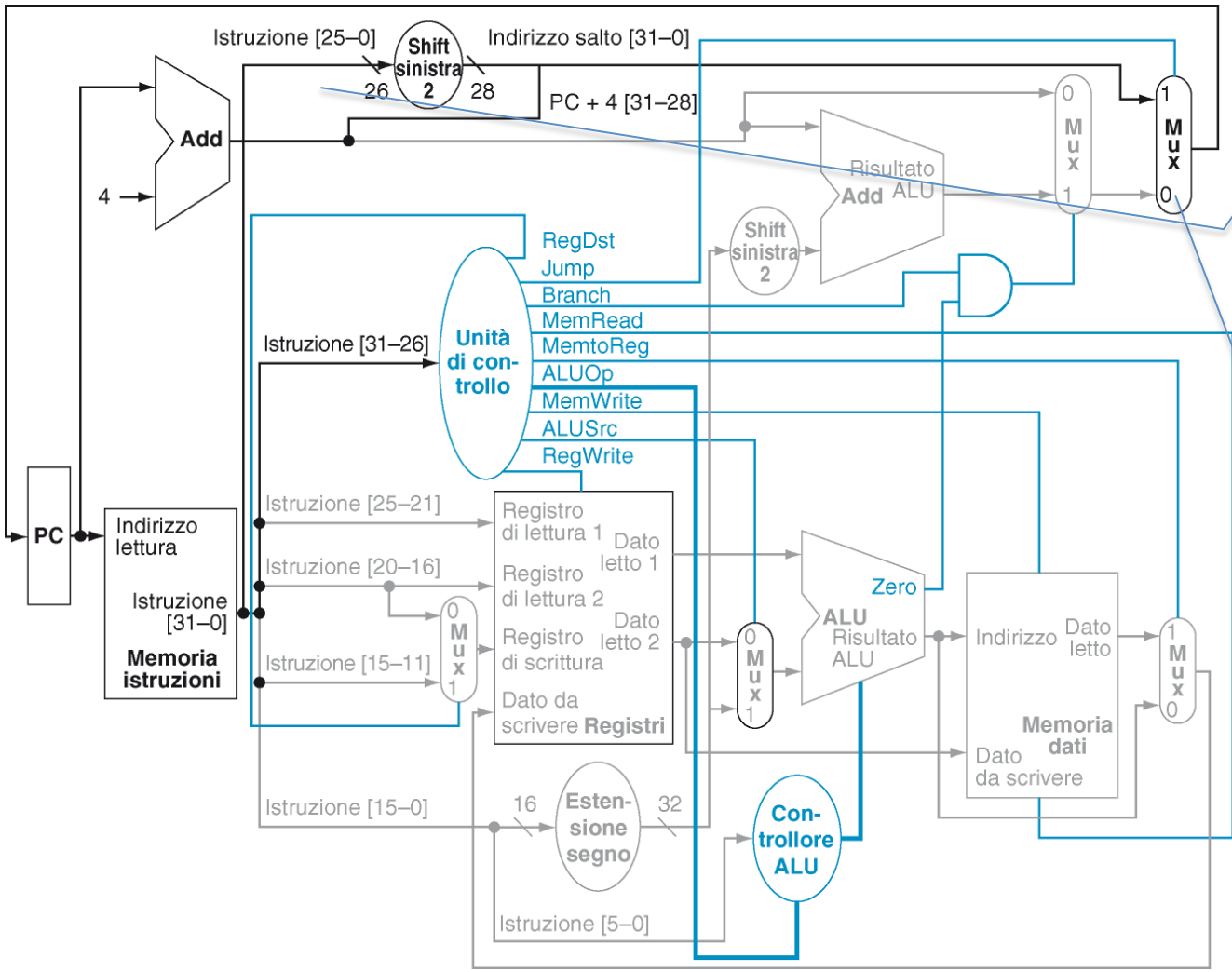
- Il formato dell'istruzione di salto incondizionato e' la seguente:

| Campo | 000010 | indirizzo |
|---|---|---|
| Posizione dei bit | 31-26 | 25:0 |

- L'esecuzione dell''istruzione prevede:
  - incremento del PC di 4
  - sostituire al PC un nuovo valore cosi' ottenuto:
    - I quattro bit + significativi rimangono uguali
    - I bit dal 27 all'1 sostituiti con il campo indirizzo dell'istruzione
    - I due bit meno significativi sono messi a zero

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Modifiche al processore




I 4 bit piu' significativi del PC vegnon giustapposti ai 26 bit del campo dinirizzi (shiftati di due

Un ulteriore multiplexer permette di settare il PC all'indirizzo del jump

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

# Considerazioni Conclusive

- Abbiamo visto come realizzare un semplice processore che esegue le istruzioni in un ciclo
- Questo non si fa piu' perche':
  - A dettare il clock sono le istruzioni piu' lente (accesso alla memoria)
  - Se si mettono istruzioni piu' complesse di quelle che abbiamo visto, le cose peggiorano ancora di piu' (esempio istruzioni floating point)
  - Non si riesce a fare ottimizzazioni aggressive sulle cose fatte piu' di frequente.