Architettura degli Elaboratori

Il Processore:
l'Unità di Elaborazione (Datapath)

Barbara Masucci
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Punto della situazione

- Abbiamo studiato
  - I circuiti combinatori e visto come funziona l'ALU
  - Un sottoinsieme dell'IS del MIPS
  - Gli elementi di base dei circuiti sequenziali e il loro utilizzo per la costruzione dei registri

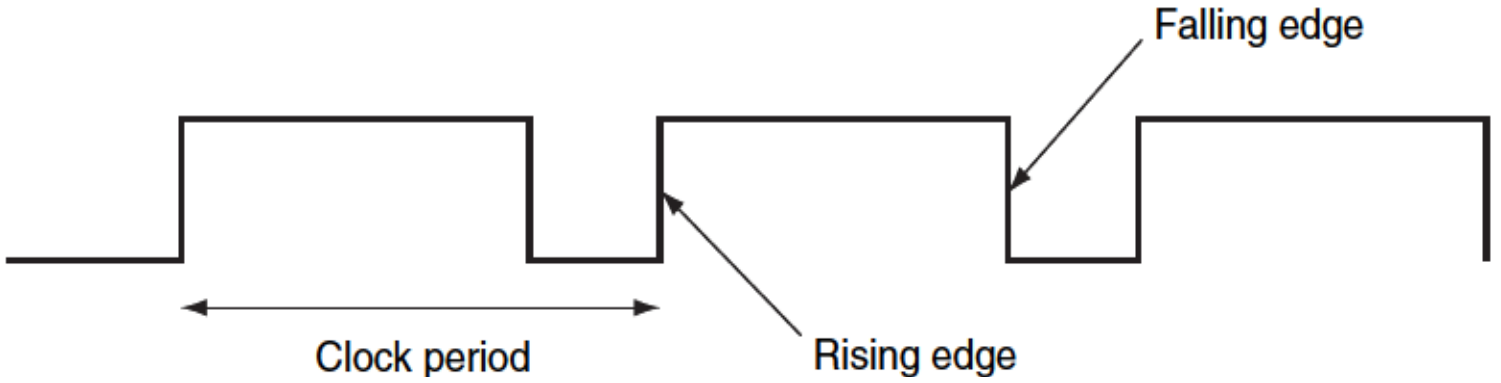
- Obiettivo di oggi e delle prossime lezioni
  - Studio del funzionamento del processore MIPS
    - Unità di Elaborazione Dati (Datapath)
    - Unità di Controllo (Control Unit)
  - Considereremo due diverse implementazioni hardware dell'IS del MIPS
    - Una implementazione di base (a ciclo singolo)
    - Una implementazione più realistica, basata su pipeline

Implementazione a Ciclo Singolo

- In questa implementazione, ciascuna istruzione impiega un ciclo singolo di clock
  - Il clock è il segnale di temporizzazione usato per sincronizzare il funzionamento dei diversi dispositivi presenti nel processore
  - Il ciclo di clock è il tempo che intercorre tra l'esecuzione di due colpi di clock successivi
  - Il periodo di clock è il tempo necessario per completare un ciclo di clock e si misura in secondi (s)
    - picosecondi: 1 ps = 10^{-12} s
    - nanosecondi: 1 ns = 10^{-9}
  - La frequenza di clock è l'inverso del periodo di clock e si misura in Hertz (cicli al secondo)
    - Mega Hertz: 1MHz = 10^6Hz = 10^6 cicli/s
    - Giga Hertz: 1GHz = 10^9Hz = 10^9 cicli/s

Implementazione a Ciclo Singolo

- In questa implementazione, ciascuna istruzione impiega un ciclo singolo di clock
  - Ogni istruzione inizia sul fronte di salita del segnale di clock
  - L'esecuzione di tutte le istruzioni ha la stessa durata (pari alla lunghezza del ciclo di clock)
  - Di conseguenza, il ciclo di clock deve essere tanto lungo da consentire l'esecuzione dell'istruzione piu' lenta



Implementazione a Ciclo Singolo

- Esamineremo una implementazione che comprende un insieme di istruzioni ridotto
  - Istruzioni di accesso alla memoria (lw, sw)
  - Istruzioni aritmetico-logiche (add, sub, and, or, slt)
  - Istruzioni di salto condizionato (beq)
- Le altre istruzioni si implementano con tecniche simili
- I concetti descritti per l'implementazione di questo set di istruzioni sono universali
  - Su di essi si basa la maggior parte dei calcolatori

Implementazione a Ciclo Singolo

- Per realizzare ciascuna istruzione dell'IS del MIPS vengono eseguite diverse fasi
- Le prime due fasi, comuni a ciascuna istruzione, sono:
  - Prelievo dell'istruzione dalla memoria che contiene il programma (fase di fetch)
  - Lettura del valore di uno o più registri operandi (estratti dai campi dell'istruzione)
- I passi successivi dipendono dalla specifica istruzione, ma sono molto simili per ciascuna delle tre classi considerate
  - Istruzioni di accesso alla memoria
  - Istruzioni aritmetico-logiche
  - Istruzioni di salto condizionato

Implementazione a Ciclo Singolo

- Tutti i tipi di istruzioni, eccetto i salti incondizionati, utilizzano l'ALU dopo aver letto i registri operandi
  - Le istruzioni di accesso alla memoria, per calcolare l'indirizzo della locazione da cui caricare o dove salvare i dati

```
lw $t0, 4($s3)
sw $t0, 32($s3)
```

  - Le istruzioni aritmetico/logiche per eseguire l'operazione corrispondente

```
add $t0, $s1, $s2
sub $t1, $s1, $s2
and $t2, $s1, $s2
```

Implementazione a Ciclo Singolo

- Dopo l'uso della ALU, il comportamento differisce per le tre classi
  - Le istruzioni di accesso alla memoria prelevano o salvano il dato in memoria
  - Le istruzioni aritmetico-logiche memorizzano il risultato nel registro destinazione
  - Le istruzioni di salto condizionato cambiano l'indirizzo dell'istruzione successiva a seconda dell'esito del confronto oggetto della condizione

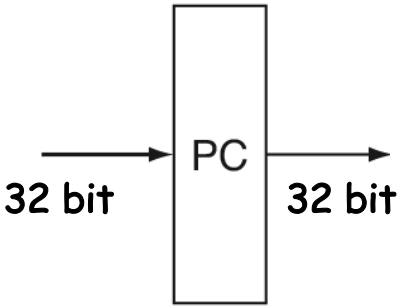
Implementazione a Ciclo Singolo

- Vediamo uno schema astratto dell'implementazione di un sottoinsieme delle istruzioni MIPS, comprendente diverse unità funzionali:
  - Program Counter (PC)
  - Memoria Istruzioni
  - Banco dei Registri
  - ALU
  - Memoria Dati

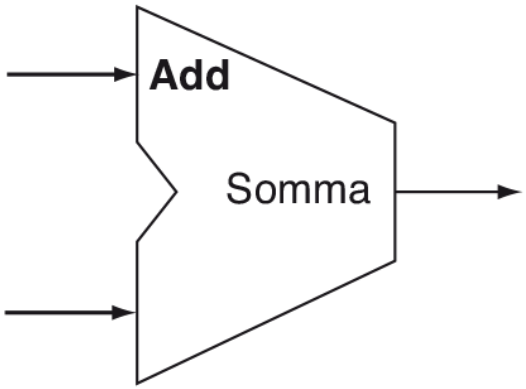
Unità funzionali: Program Counter

Il Program Counter (PC)

- E' un registro a 32 bit contenente l'indirizzo dell'istruzione corrente



- Dopo l'esecuzione dell'istruzione, deve essere aggiornato (al termine del ciclo di clock) in modo che punti alla prossima istruzione da eseguire
  - Ciò può essere fatto incrementando di 4 il suo contenuto mediante un opportuno sommatore (Add)
  - Il sommatore è un circuito combinatorio costruito a partire dalla ALU, impostando i suoi segnali di controllo in modo che esegua sempre una somma



Unità funzionali: Memoria Istruzioni

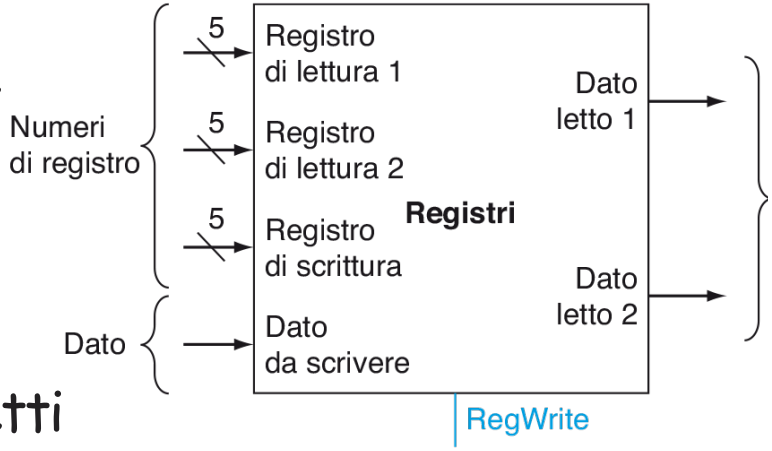
La Memoria Istruzioni

- Fornisce solo un accesso in lettura perché non viene scritta dall'Unità di Elaborazione Dati
  - In realtà viene scritta solo in fase di caricamento del programma
- Dato in input l'indirizzo di lettura, fornisce il contenuto (istruzione a 32 bit) della locazione specificata

Unità funzionali: Banco dei Registri

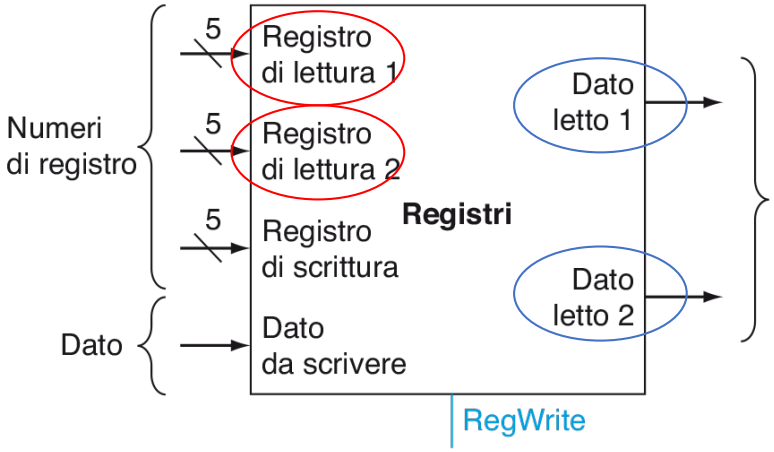
Il Banco dei Registri

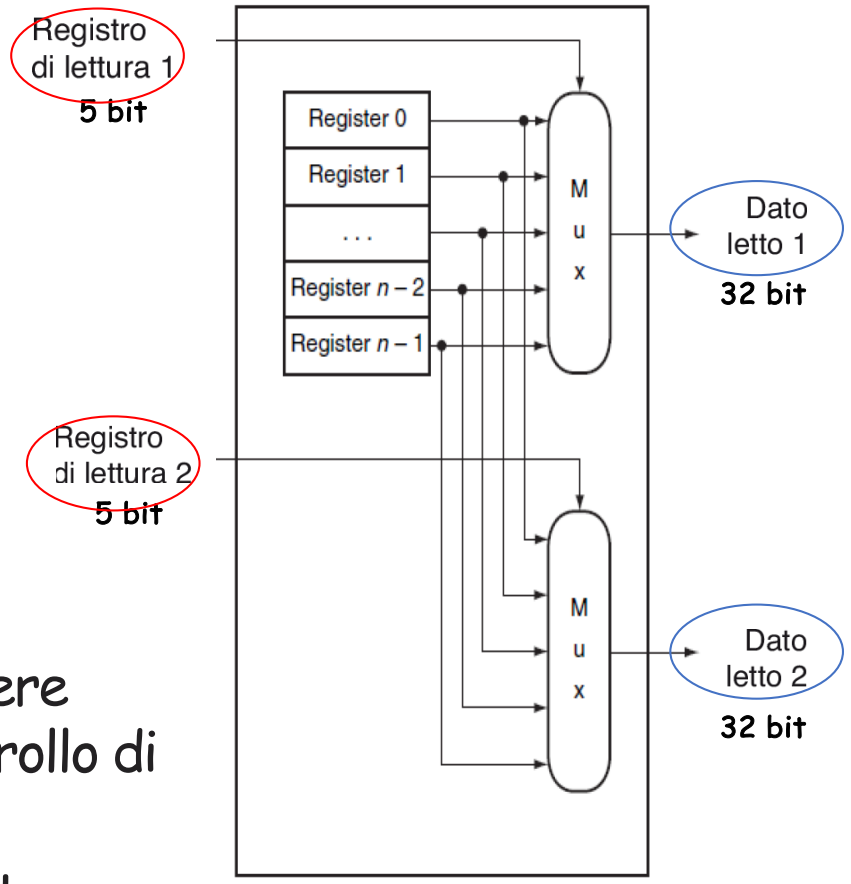
- E' un insieme di registri che possono essere letti o scritti
- Input:
  - I numeri dei due registri da leggere
  - Il numero del registro da scrivere
  - Il dato da scrivere
- Output:
  - I contenuti (dati) dei due registri letti



- La scrittura di un dato in ingresso viene controllata dal segnale RegWrite che deve essere posto a 1 per poter scrivere nel registro

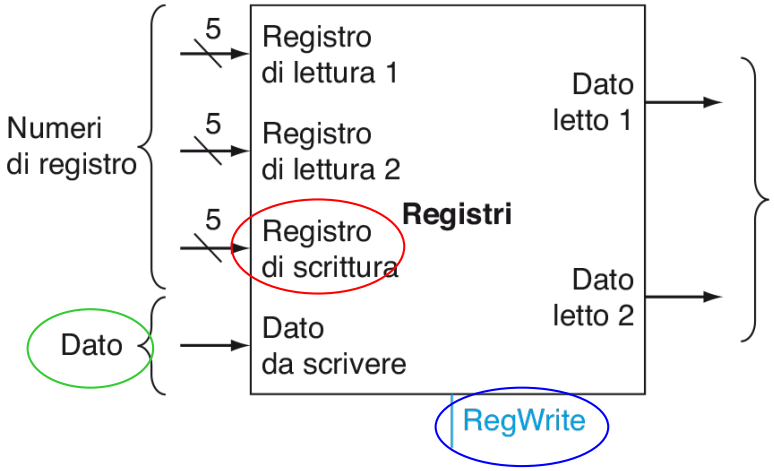
Unità funzionali: Banco dei Registri

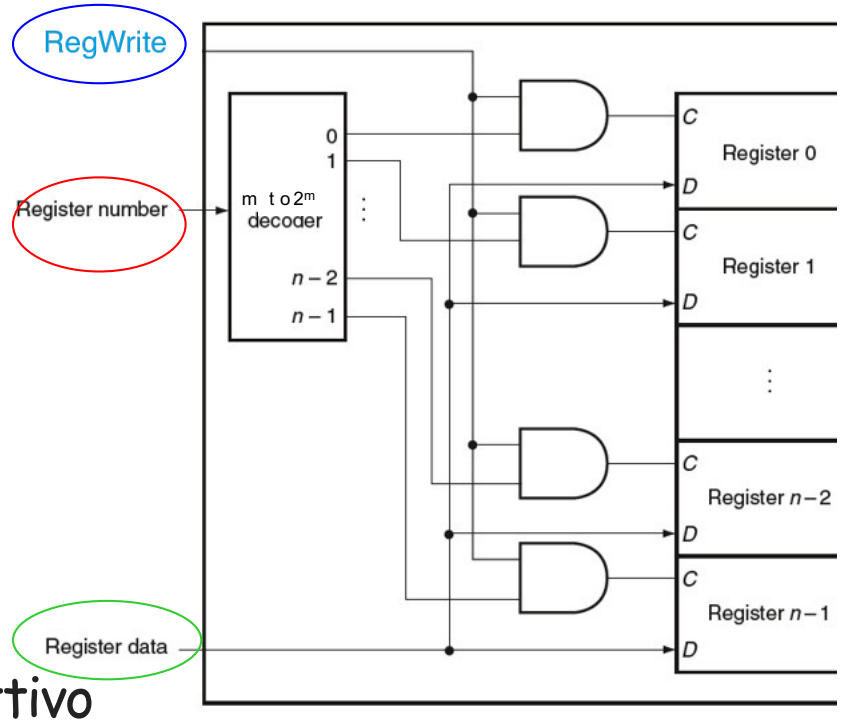




- Lettura
  - I numeri dei due registri da leggere (5 bit) diventano i segnali di controllo di due MUX 2^5:1
  - Ciascun MUX fornisce in output il contenuto del registro corrispondente (32 bit)

Unità funzionali: Banco dei Registri



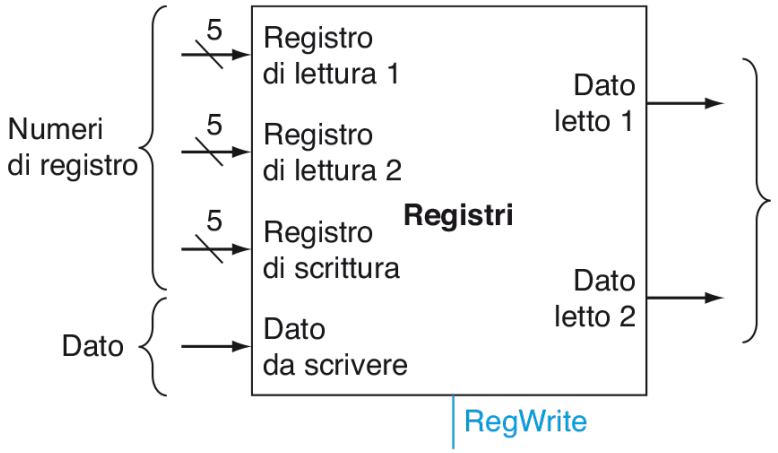


- Scrittura
  - Il segnale RegWrite deve essere attivo
  - Il numero del registro da scrivere (5 bit) va in input a un decoder 5-to-32 che seleziona il registro da scrivere
  - Il dato da scrivere (32 bit) va in input ai 32 flip-flop D che costituiscono quel registro (il cui stato viene modificato)

Unità funzionali: Banco dei Registri

Nel Banco dei Registri

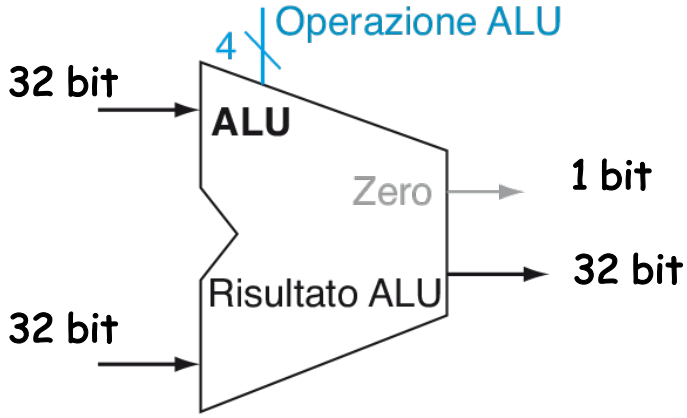
- Si può contemporaneamente leggere e scrivere in uno stesso registro nello stesso ciclo di clock
  - La lettura fornisce il dato scritto al ciclo di clock precedente
  - Il valore scritto potrà essere letto al ciclo di clock successivo



Unità funzionali: ALU

L' ALU

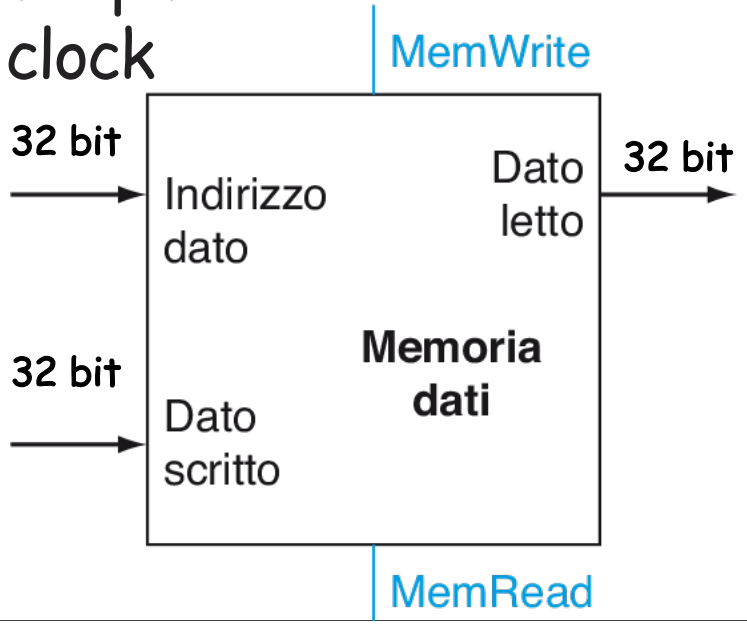
- Ha due input a 32 bit e fornisce un risultato a 32 bit
- Inoltre fornisce in output il bit Zero
  - Zero=1 se il risultato dell'operazione è 0
- Ha un segnale di controllo di 4 bit (Operazione ALU), per selezionare l'operazione da eseguire



Unità funzionali: Memoria Dati

La Memoria Dati

- Ha come ingresso l'indirizzo del dato e il dato da scrivere
- Possiede una sola uscita per il dato letto
- E' gestita da due segnali di controllo separati per la lettura e la scrittura, di cui solo uno può essere posto a 1 all'interno di un ciclo di clock
  - MemWrite
  - MemRead

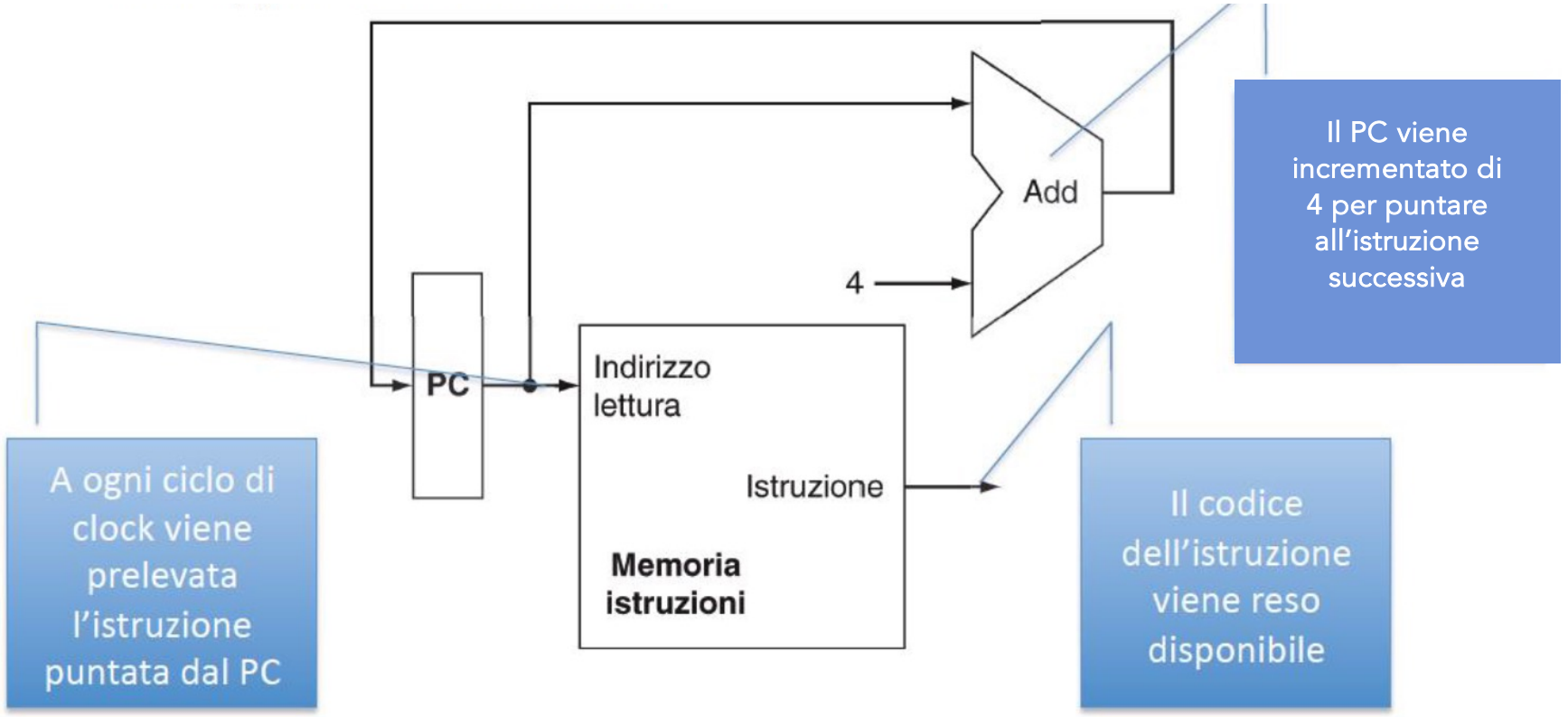


Implementazione a ciclo singolo: Prelievo istruzione

- Vediamo in dettaglio come avviene il prelievo dell'istruzione (fase di fetch)
  - Innanzitutto viene letto il contenuto del registro PC, e viene prelevata, in memoria istruzioni, l'istruzione da eseguire
  - Poi, viene aggiornato il contenuto del registro PC, in modo che punti alla prossima istruzione da eseguire

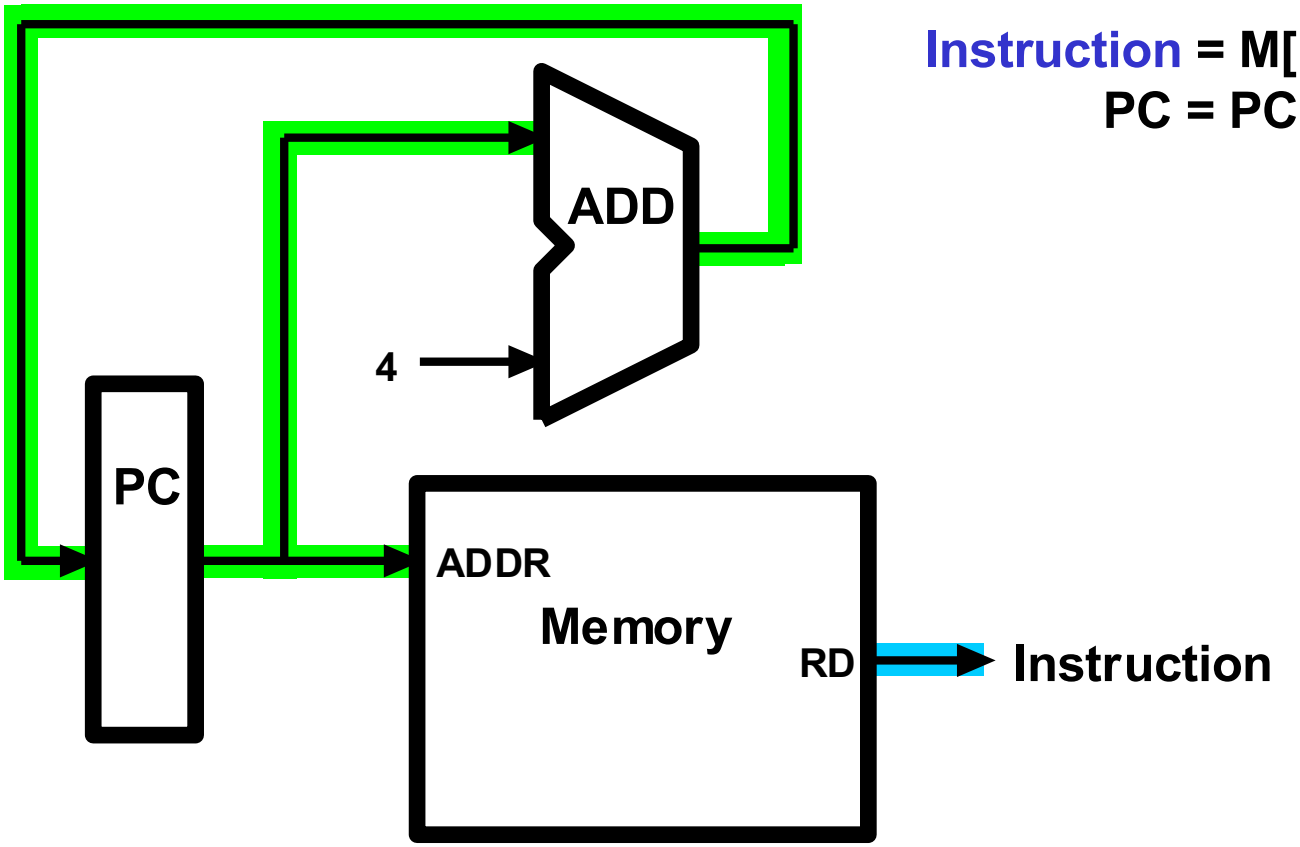
Implementazione a ciclo singolo: Prelievo istruzione

Il prelievo dell'istruzione viene effettuato con un'apposita circuiteria



Implementazione a ciclo singolo: Prelievo istruzione

Datapath nella fase di fetch



Instruction = M[PC]
PC = PC + 4

Implementazione a ciclo singolo: Istruzioni in formato R

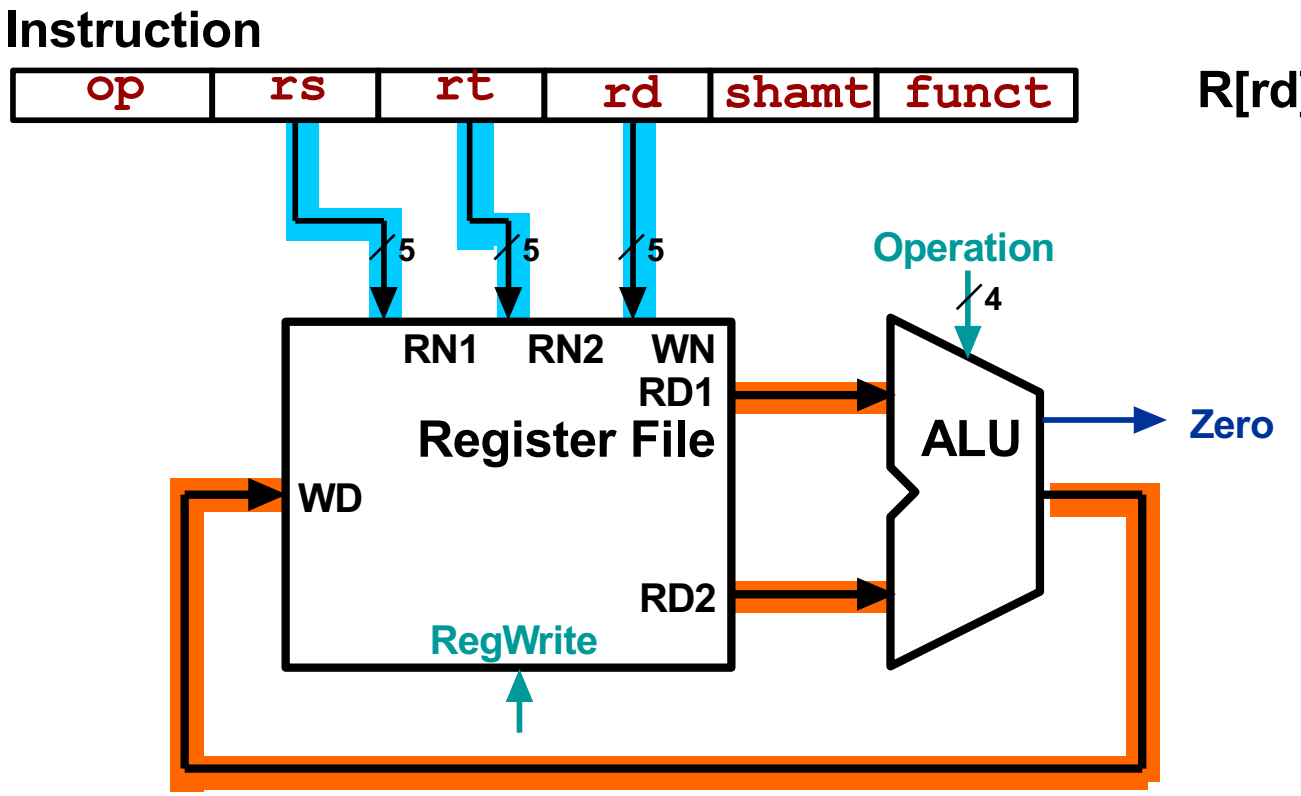
- Adesso vediamo come vengono eseguite le istruzioni in formato R
- Si tratta di istruzioni aritmetico/logiche che operano tra registri e producono un risultato che viene memorizzato in un registro
- Ad esempio, add $t0, $s1, $s2 che in formato R è

Implementazione a ciclo singolo: Istruzioni in formato R

Datapath delle istruzioni in formato R

```
add rd, rs, rt
```

R[rd] =R[rs] + R[rt];



Implementazione a ciclo singolo: Istruzioni in formato I

- Adesso vediamo come vengono eseguite le istruzioni in formato I
- Iniziamo considerando le istruzioni load e store

  lw $t0, offset($s3)

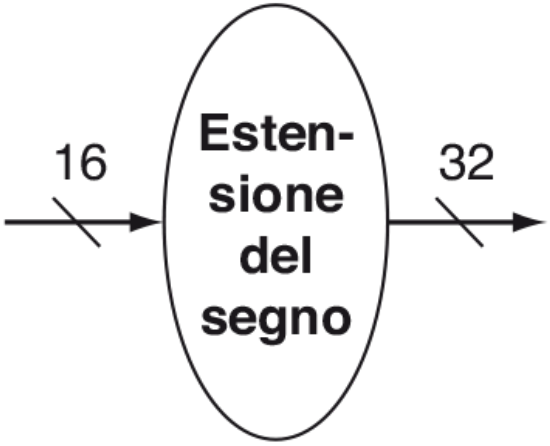
  sw $t0, offset($s3)

  il cui formato I è

- Per entrambe si deve calcolare un indirizzo di memoria dato dalla somma del contenuto del registro base con l'offset (contenuto nel campo a 16 bit)
  - Sono quindi richiesti i moduli funzionali ALU e banco dei registri

Implementazione a ciclo singolo: Istruzioni in formato I

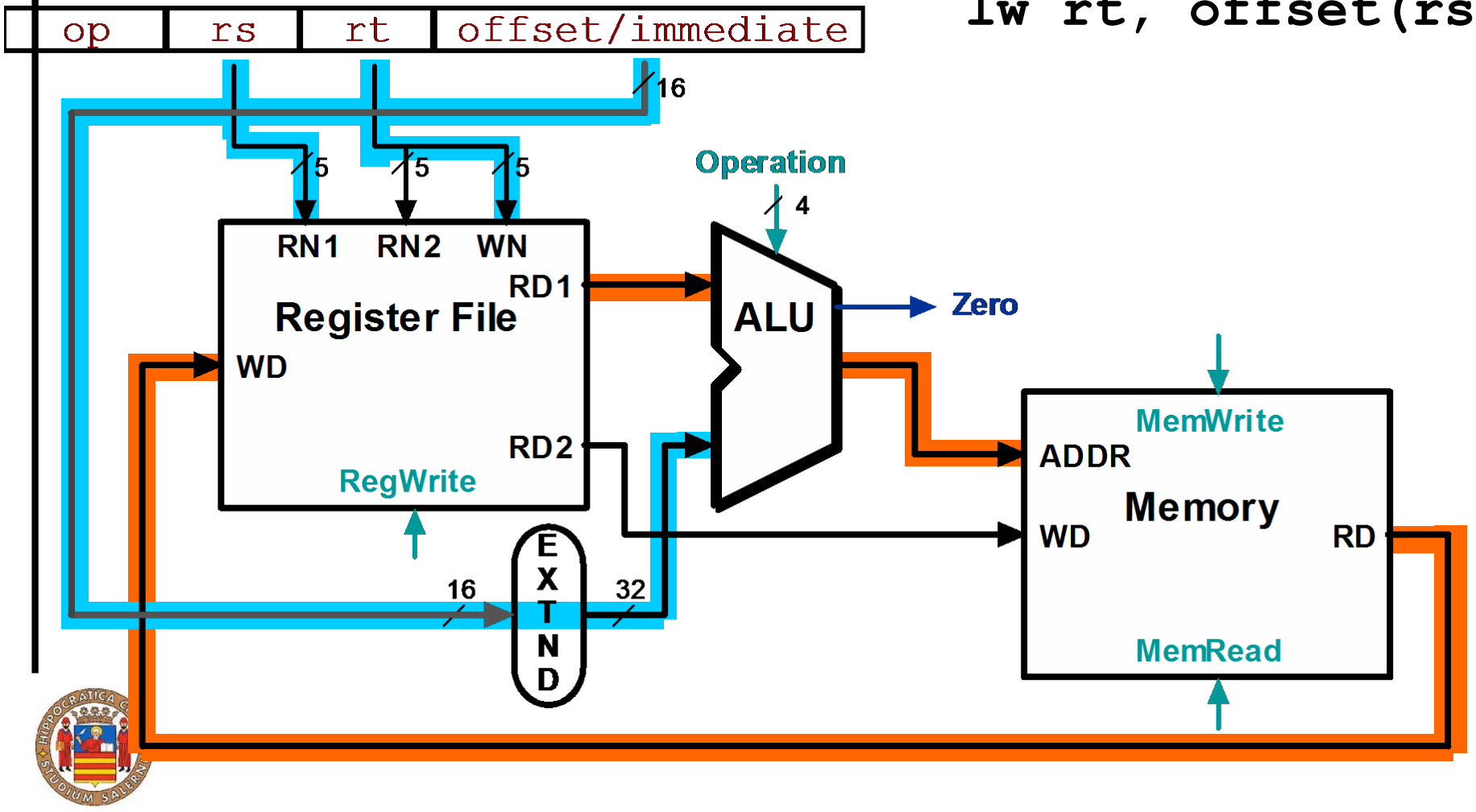
- Per effettuare la somma dell'indirizzo contenuto nel registro di base (32 bit) con l'offset (16 bit) , viene utilizzata una unità per l'estensione del segno, in modo da avere un offset a 32 bit
  - Replica il bit più significativo per 16 volte



Implementazione a ciclo singolo: Istruzioni in formato I

Datapath dell' istruzione lw

```
lw rt, offset(rs)
```

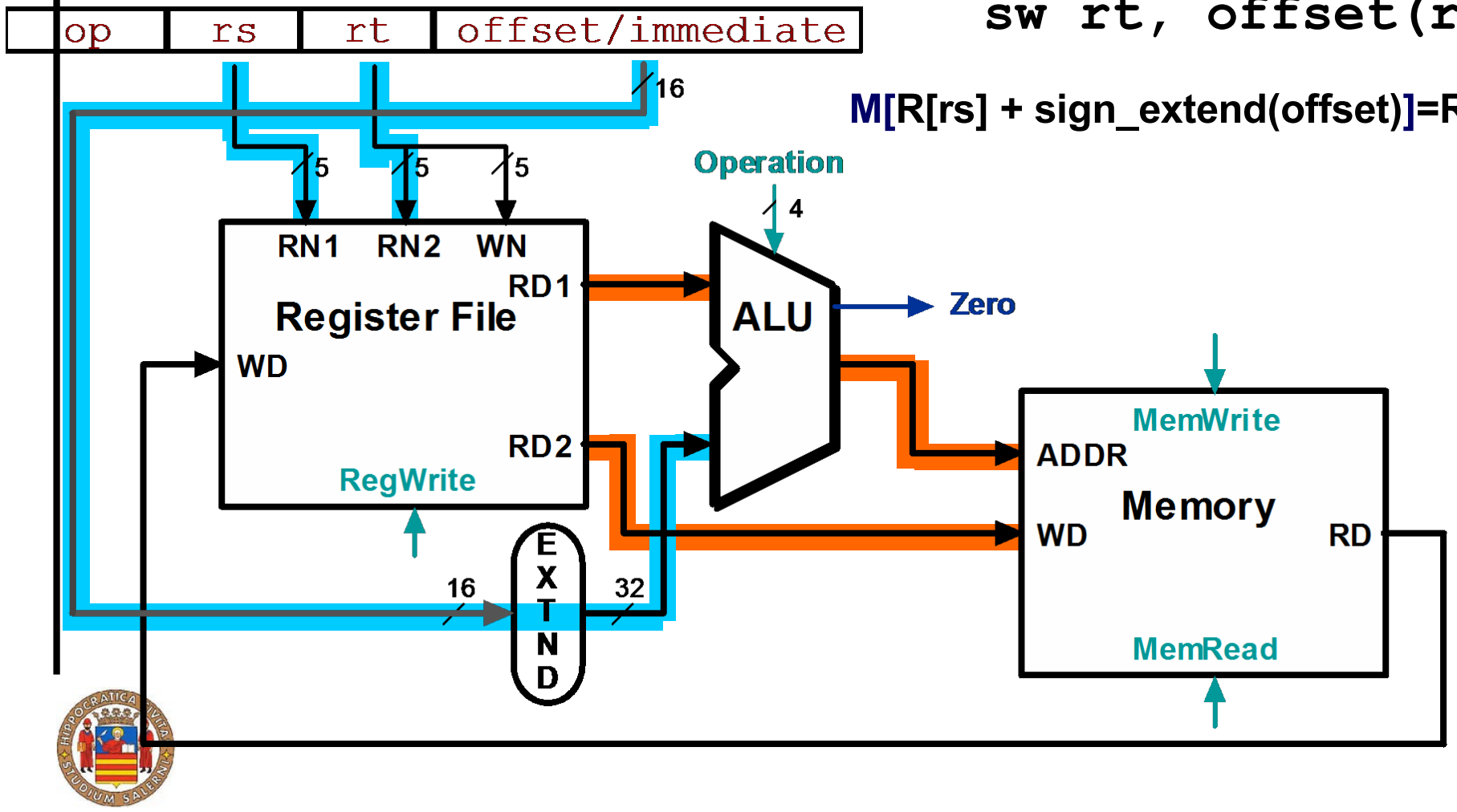


Implementazione a ciclo singolo: Istruzioni in formato I

Datapath dell' istruzione sw

```
sw rt, offset(rs)
```

M[R[rs] + sign_extend(offset)]=R[rt]



Implementazione a ciclo singolo: Istruzioni in formato I

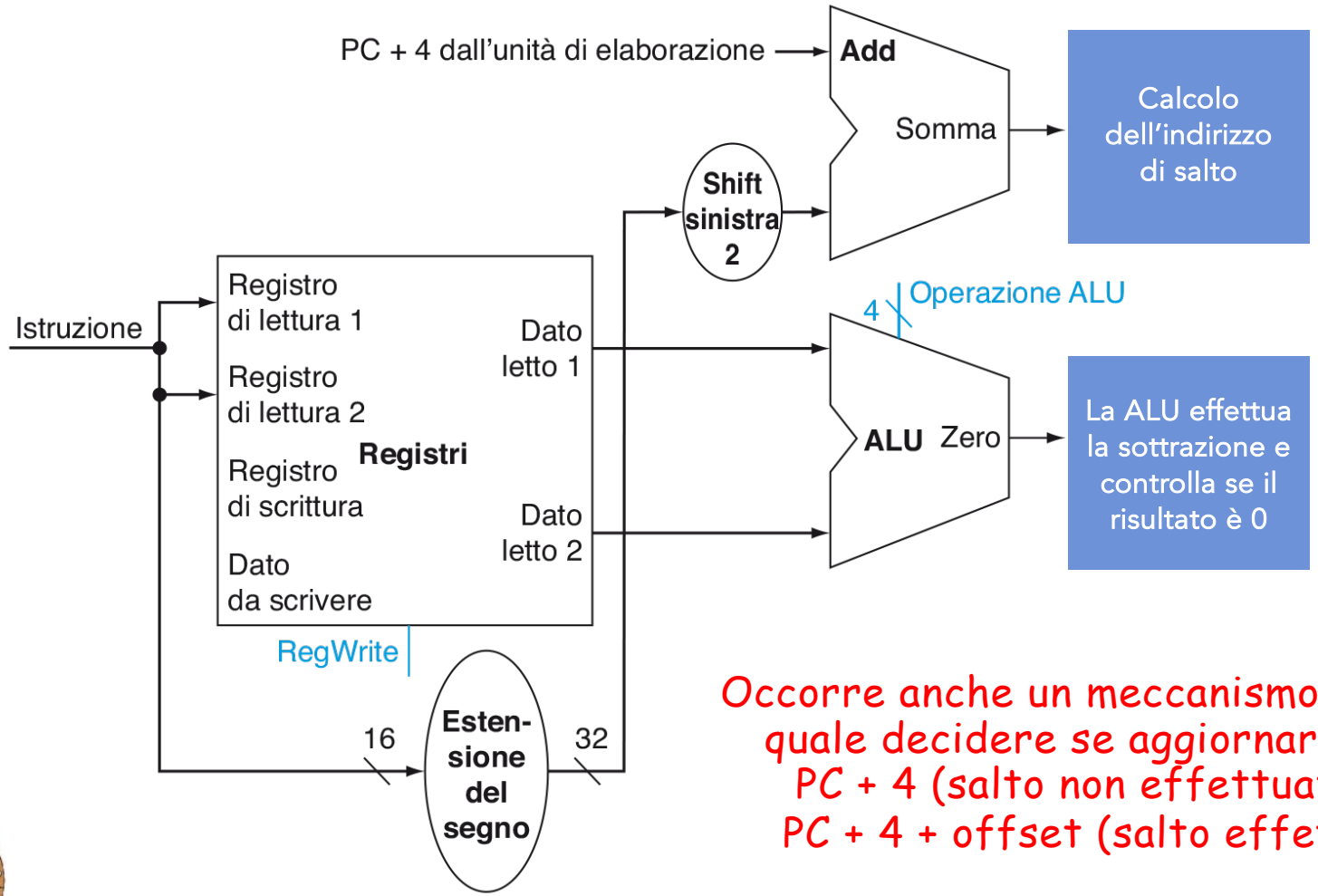
- Adesso vediamo come viene eseguita l'istruzione di salto condizionato

  beq $t1, $t2, offset

  il cui formato I è

- Se i due operandi $t1 e $t2 sono uguali
  - Viene calcolato l'indirizzo di destinazione del salto, sommando all'indirizzo della prossima istruzione (PC+4) il valore dell'offset, dopo averlo esteso a 32 bit e shiftato a sx di 2 posti

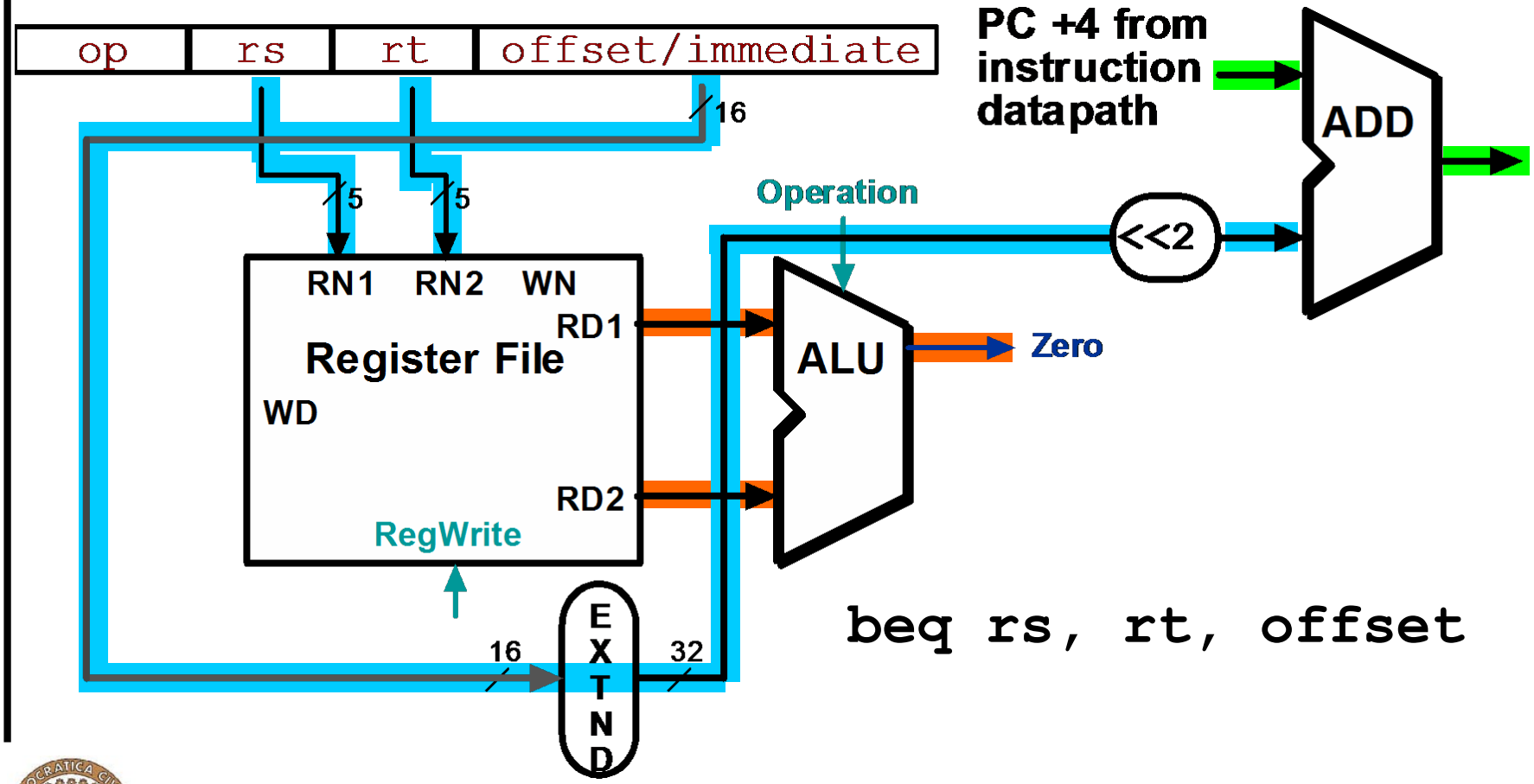
Implementazione a ciclo singolo: Istruzioni in formato I



Occorre anche un meccanismo in base al quale decidere se aggiornare il PC a PC + 4 (salto non effettuato) o a PC + 4 + offset (salto effettuato)

Implementazione a ciclo singolo: Istruzioni in formato I

Datapath dell' istruzione beq




```
beq rs, rt, offset
```

Implementazione a Ciclo Singolo

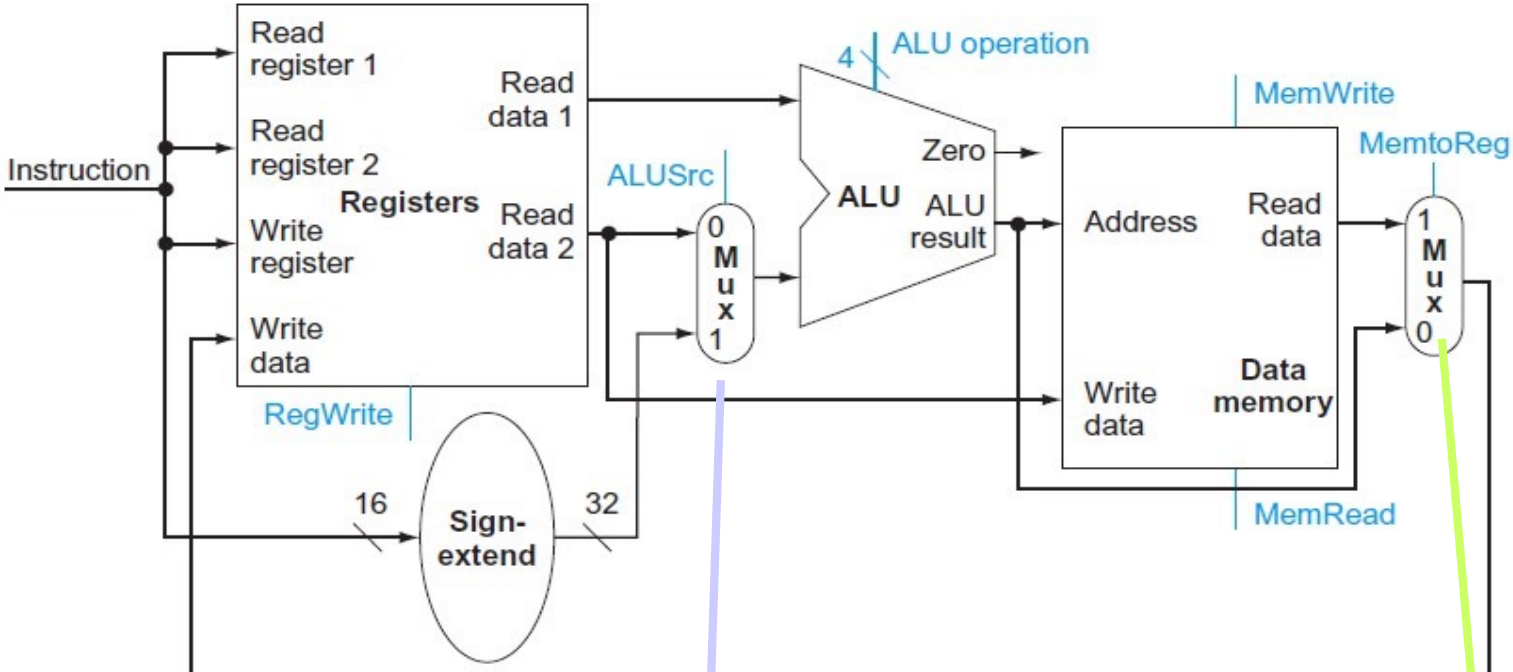
- Adesso combiniamo tra loro i vari blocchi e aggiungiamo i segnali di controllo opportuni

- Innanzitutto, notiamo che ciascuna unità funzionale può ricevere dati da diverse sorgenti:
  - Il secondo input della ALU può provenire da un registro (tipo R) o dal campo immediato di un'istruzione, esteso a 32 bit (tipo I)
  - Il dato scritto nel banco dei registri può provenire dalla ALU o dalla memoria

- E' necessario utilizzare dei multiplexer per selezionare da quale, tra le differenti sorgenti collegate, debba essere preso il dato

Implementazione a Ciclo Singolo

Uniamo il blocco relativo alle istruzioni di accesso alla memoria con quello per le istruzioni di tipo R, introducendo due multiplexer



Multiplexer per scegliere se il secondo operando della ALU è il dato in un registro (tipo R) oppure un indirizzo (tipo I)

Multiplexer per scegliere se il dato da memorizzare nel banco dei registri è il dato prelevato in memoria (tipo I) oppure il risultato dell'operazione effettuata dall'ALU (tipo R)

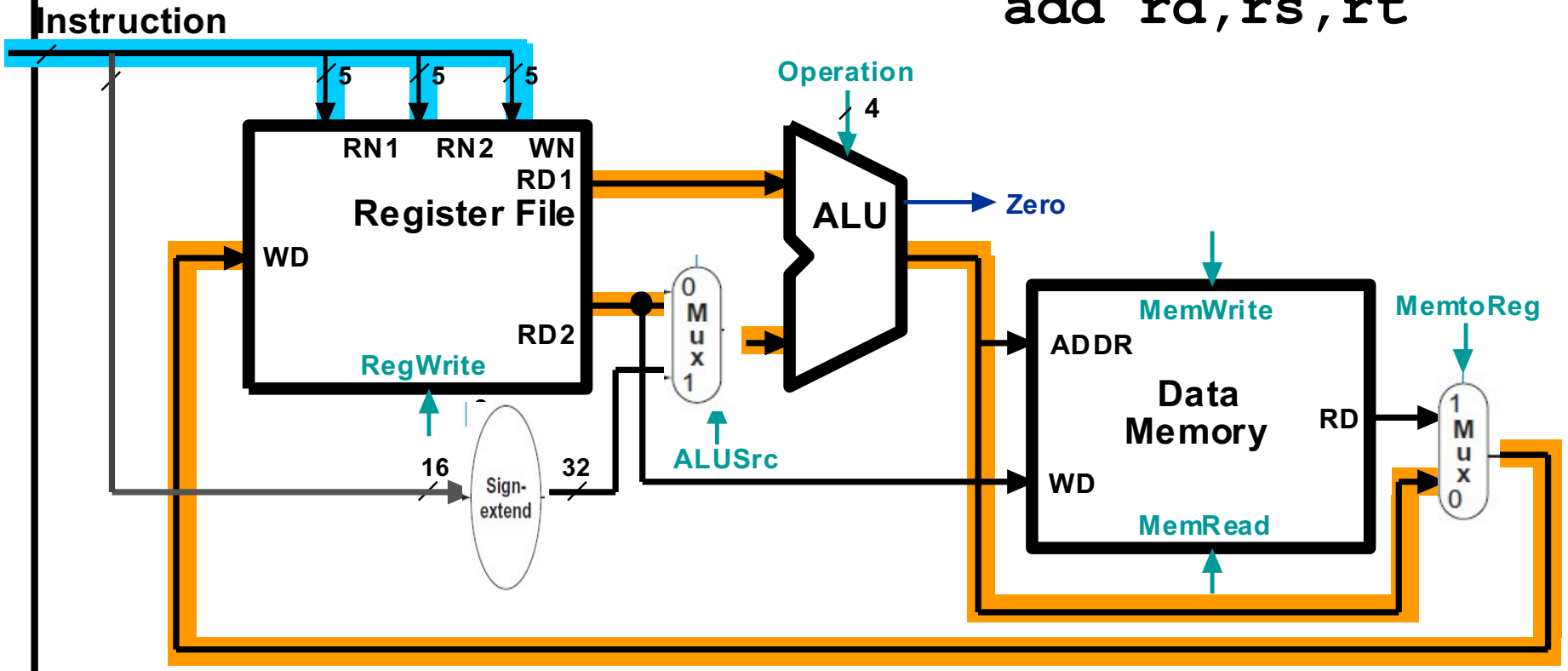
Implementazione a Ciclo Singolo

- Vediamo il datapath all'interno del circuito così ottenuto, per ciascuna delle operazioni
  - add
  - lw
  - sw

Implementazione a Ciclo Singolo

Datapath delle istruzioni in formato R

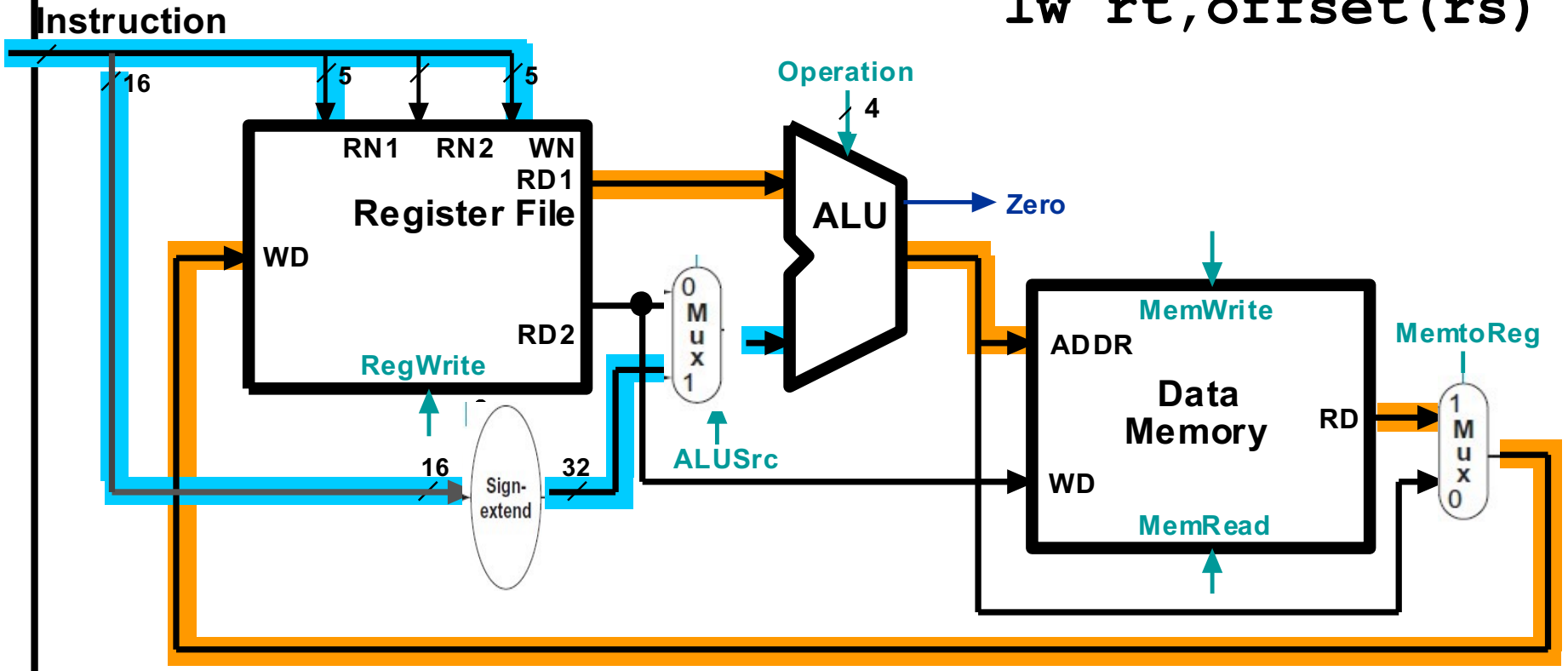
```
add rd,rs,rt
```



Implementazione a Ciclo Singolo

Datapath dell' istruzione lw

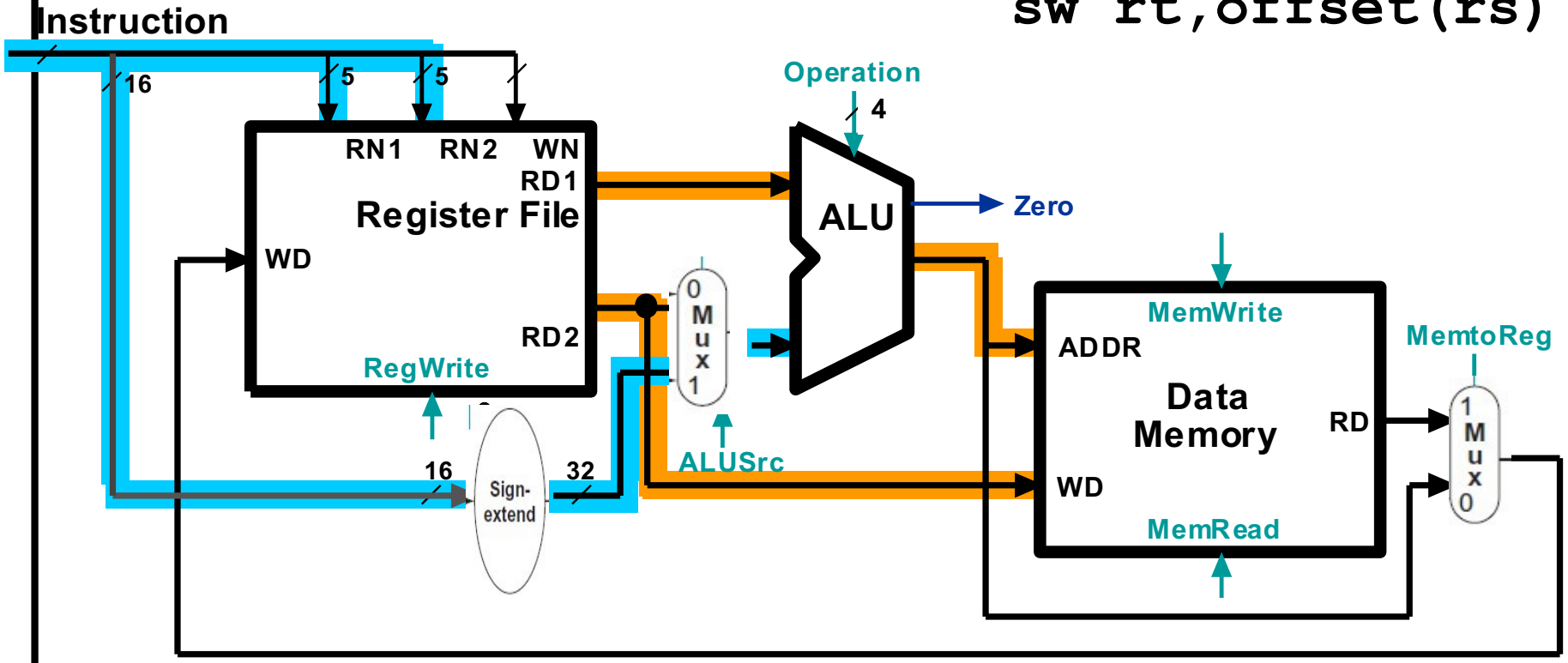
```
lw rt,offset(rs)
```



Implementazione a Ciclo Singolo

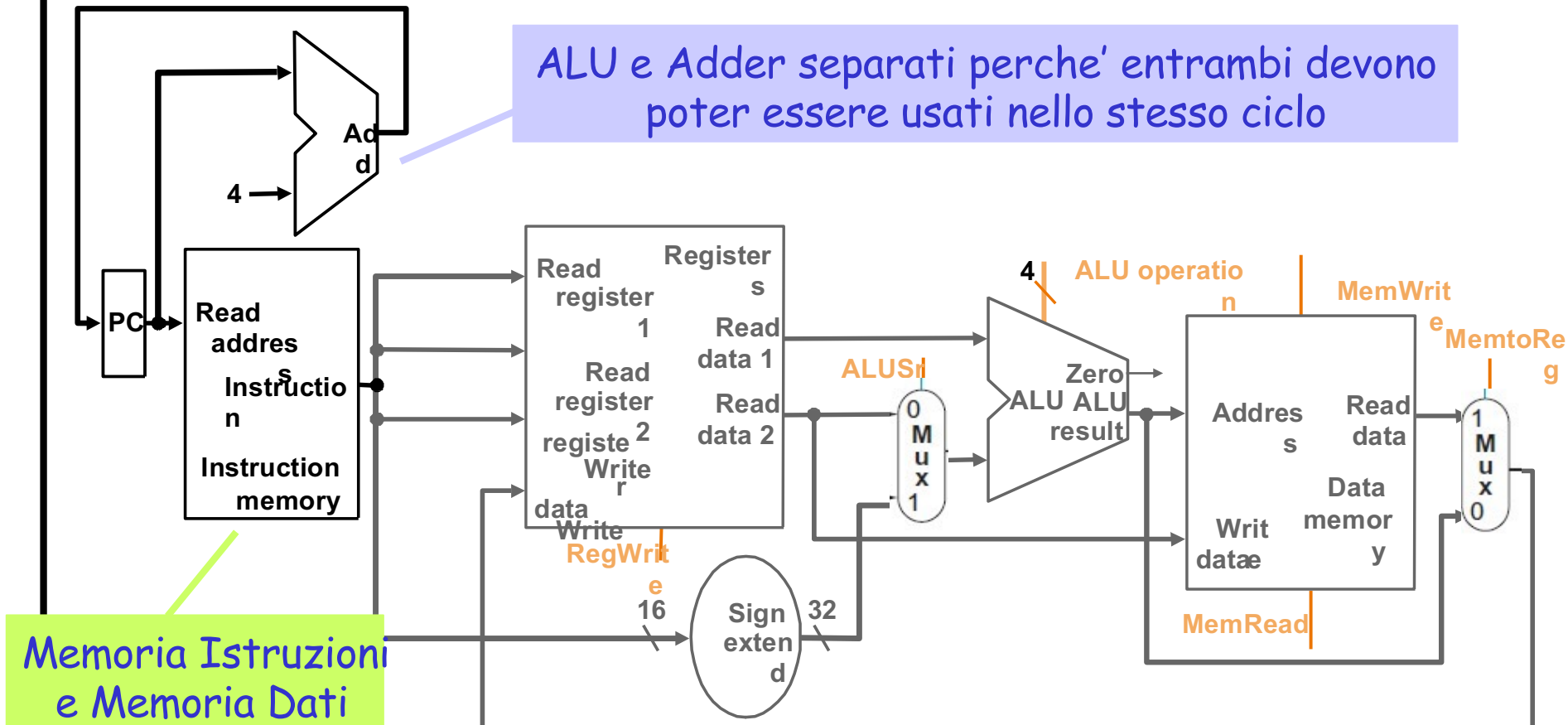
Datapath dell' istruzione sw

```
sw rt,offset(rs)
```



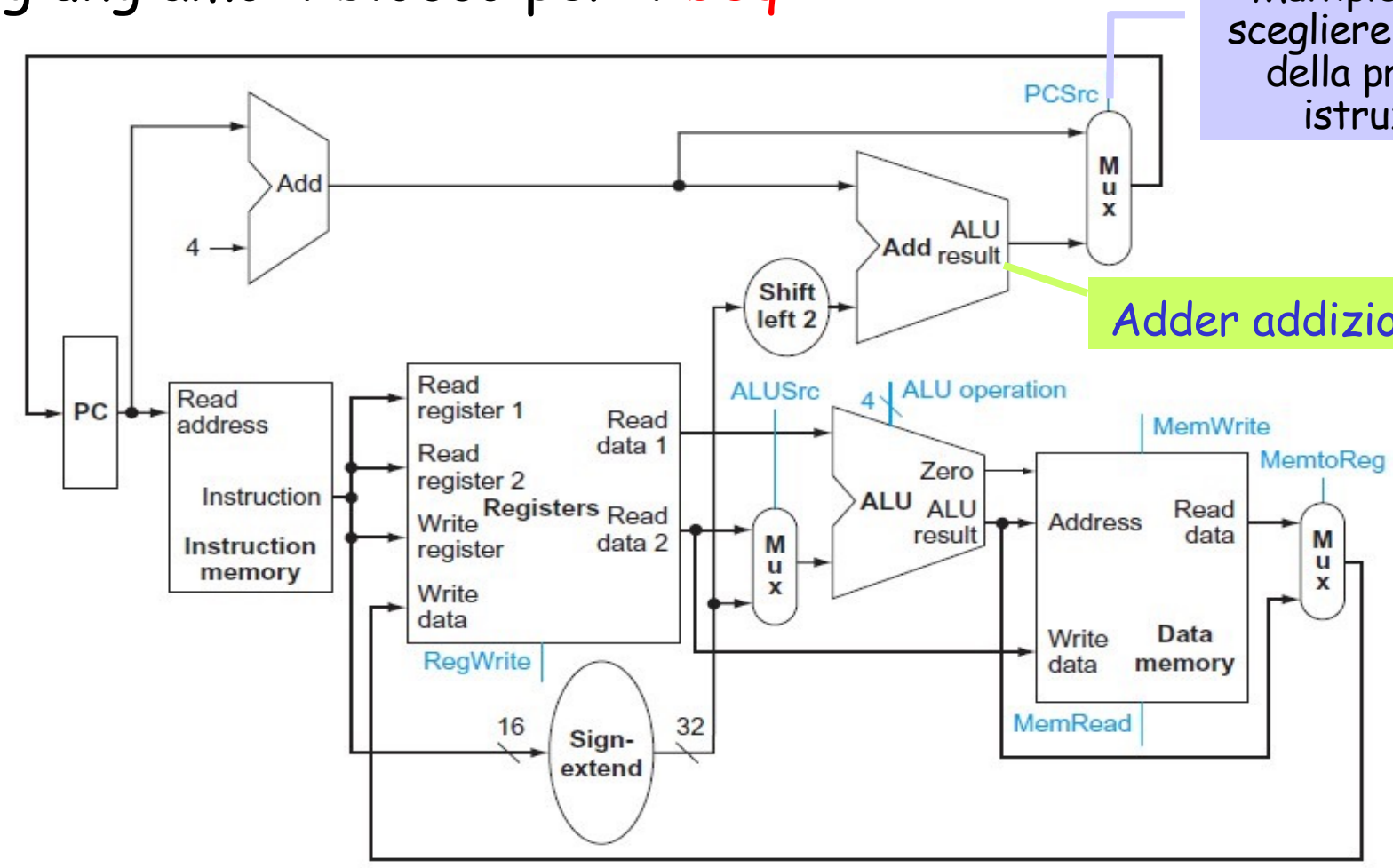
Implementazione a Ciclo Singolo

Aggiungiamo il blocco che esegue il prelievo dell'istruzione



Implementazione a Ciclo Singolo

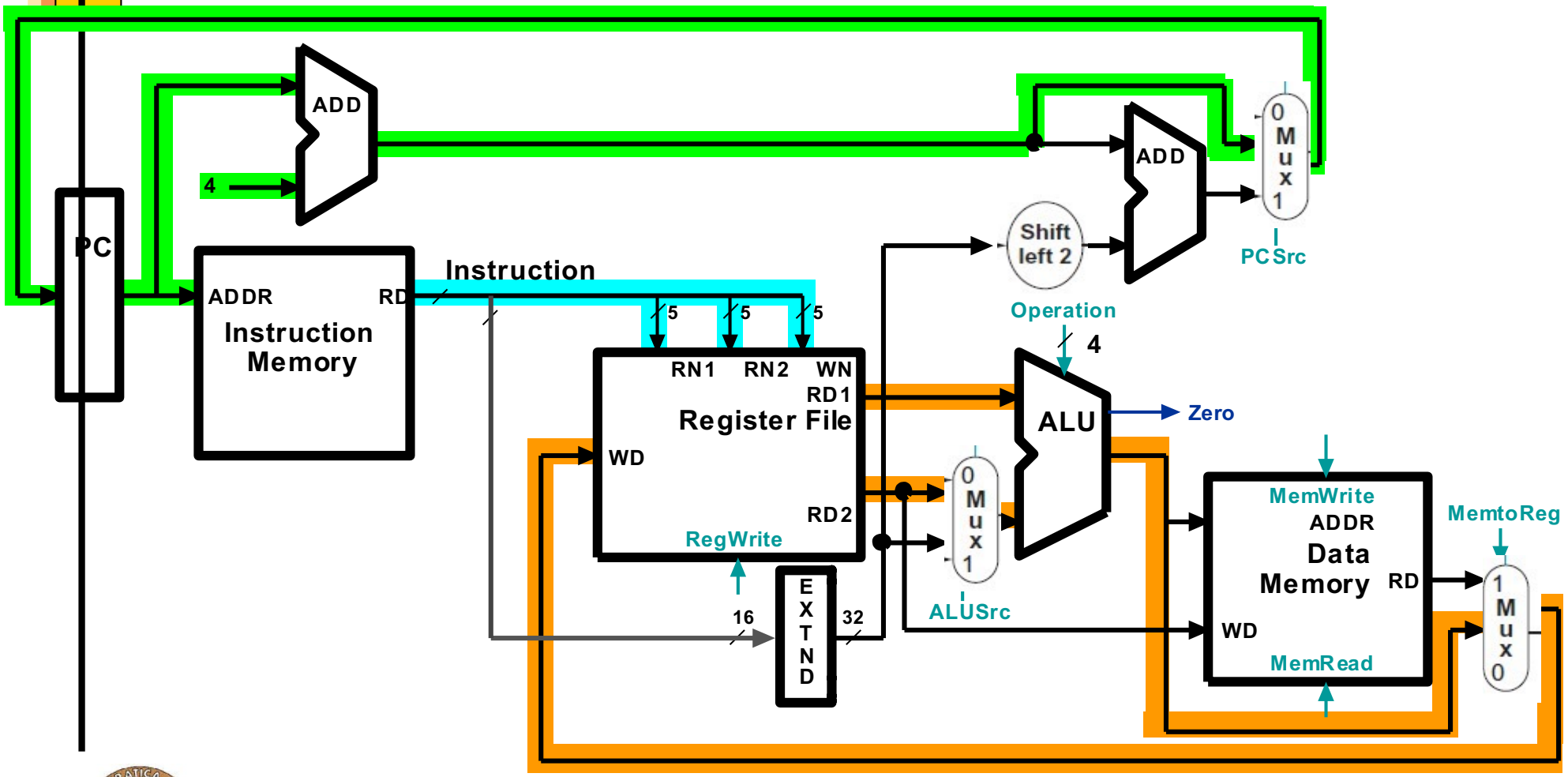
Aggiungiamo il blocco per il beq



Implementazione a Ciclo Singolo

- Vediamo il datapath all'interno del circuito così ottenuto, per ciascuna delle operazioni
  - add
  - lw
  - sw
  - beq

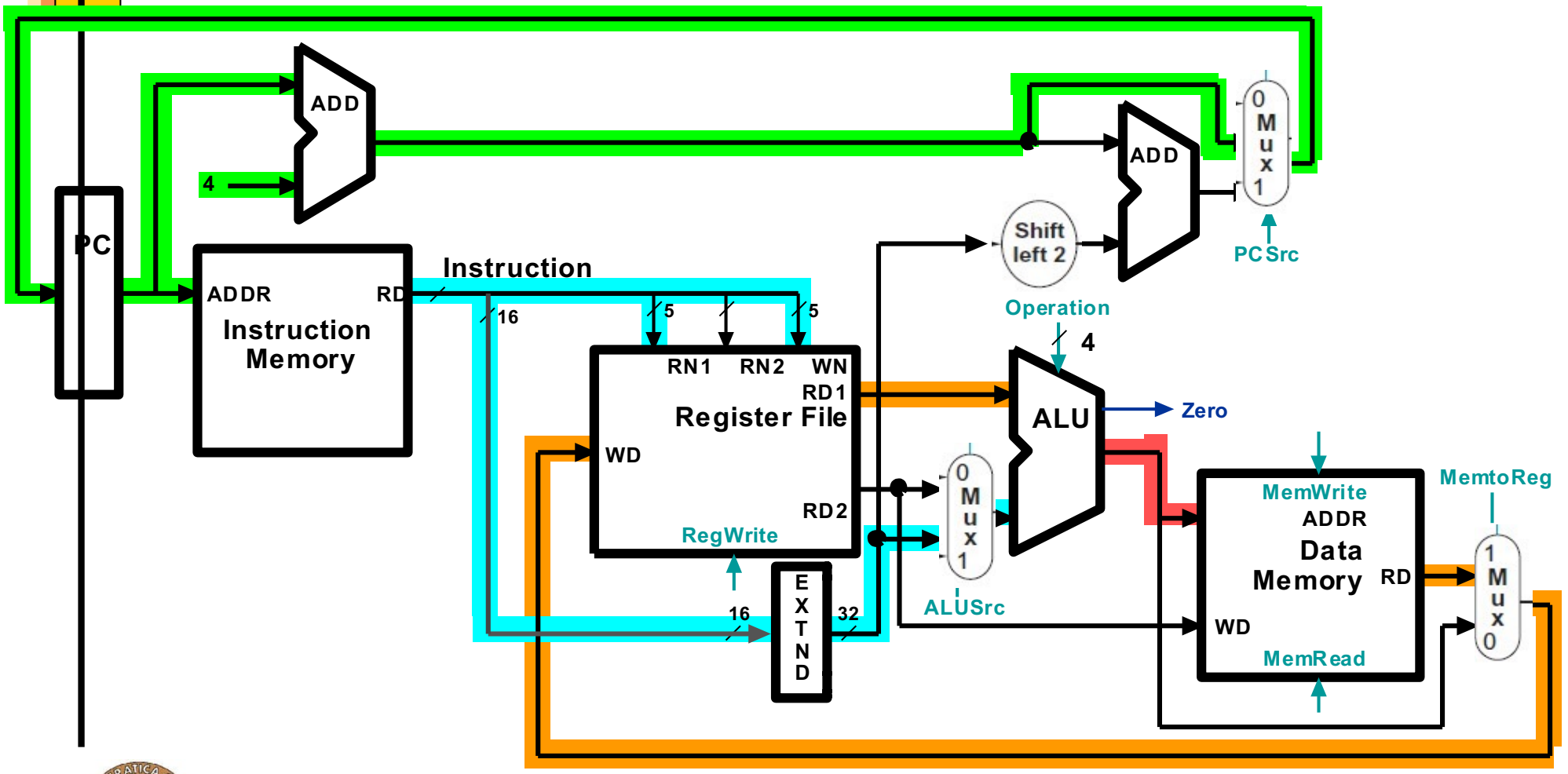
Implementazione a Ciclo Singolo



Datapath dell' istruzione add

`add rd, rs, rt`

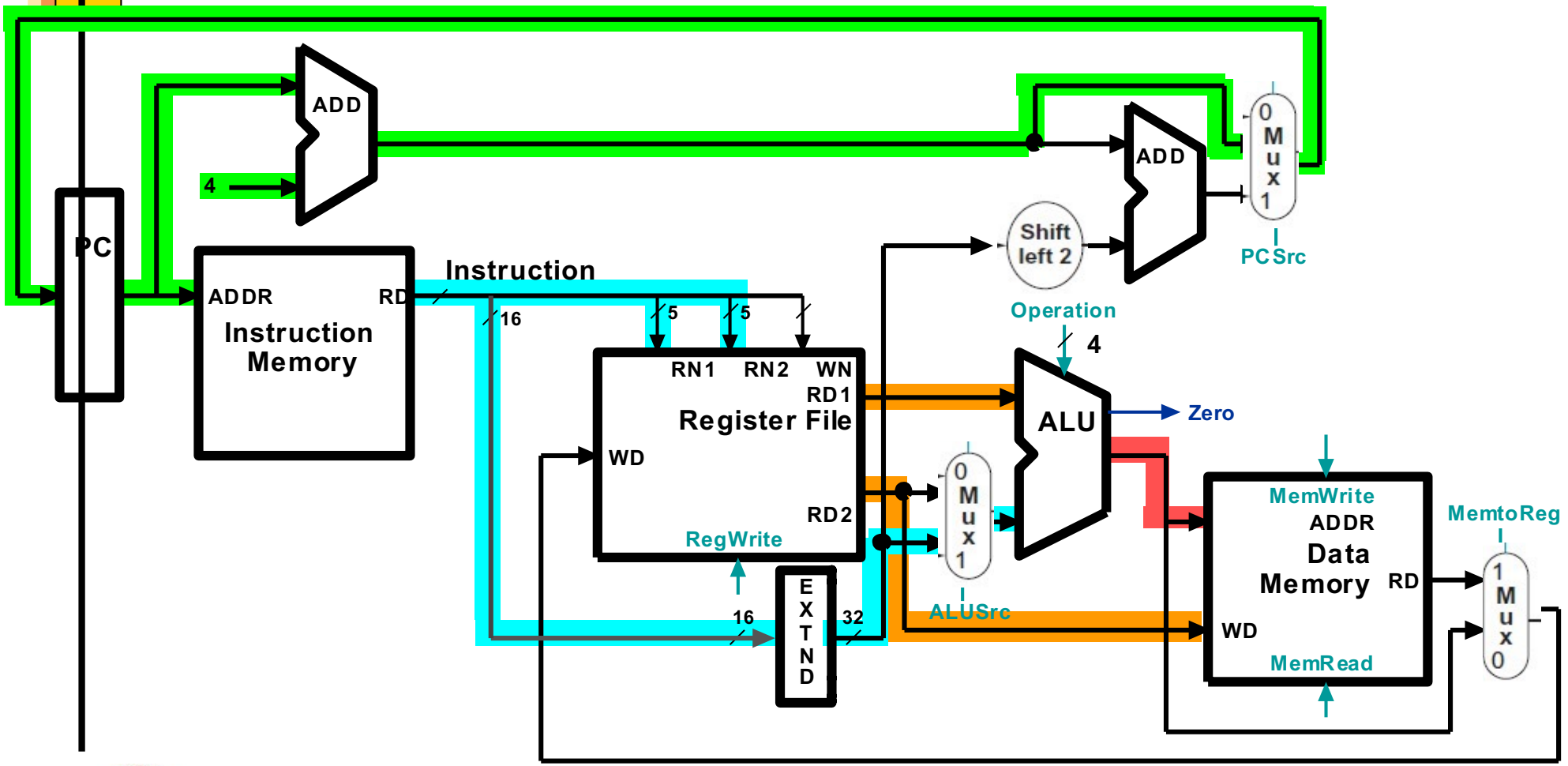
Implementazione a Ciclo Singolo



Datapath dell' istruzione lw

`lw rt,offset(rs)`

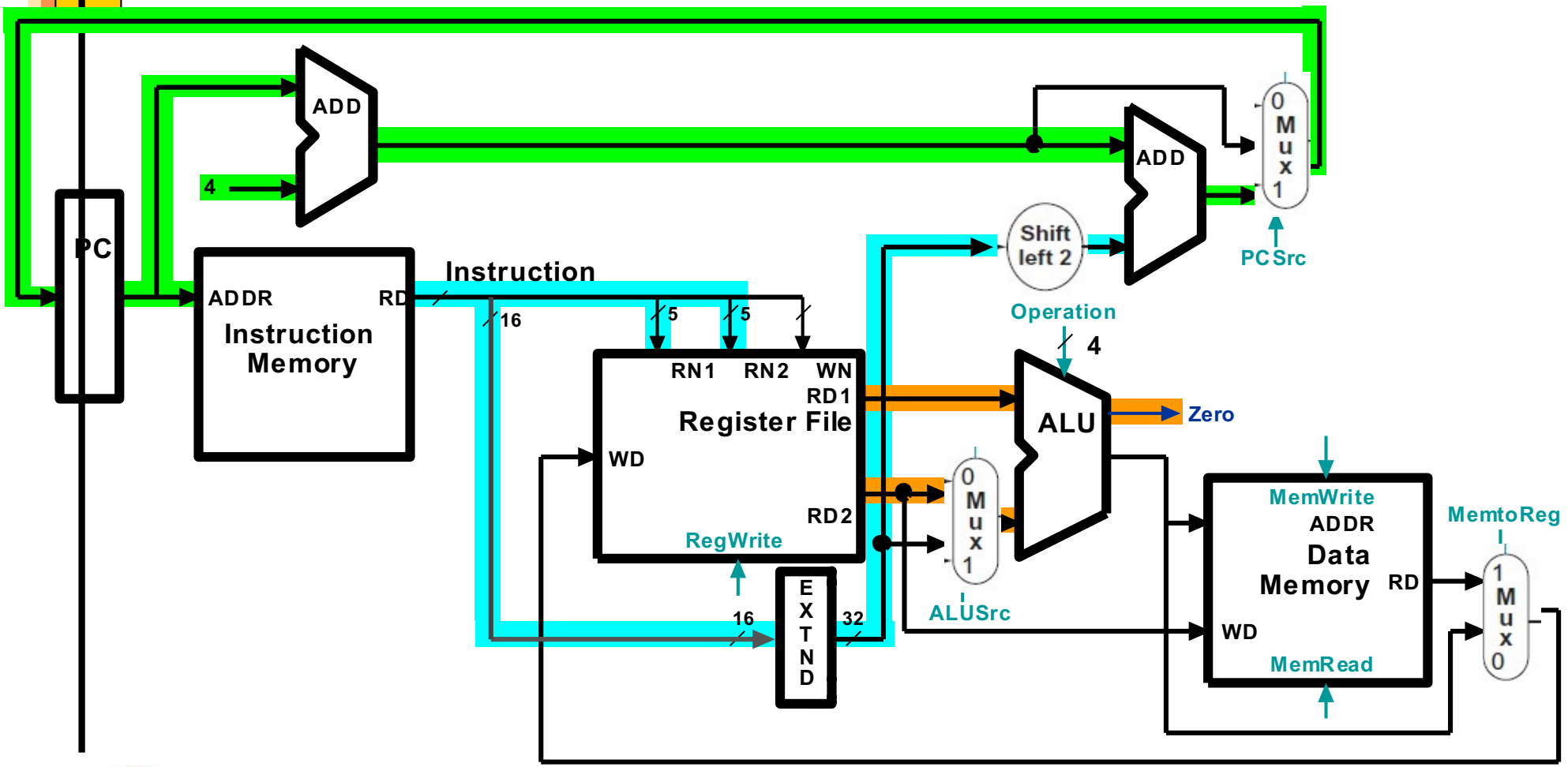
Implementazione a Ciclo Singolo



Datapath dell' istruzione sw

`sw rt,offset(rs)`

Implementazione a Ciclo Singolo



Datapath dell' istruzione beq

`beq r1,r2,offset`

Riepilogo e riferimenti

- Abbiamo studiato una prima implementazione hardware (a ciclo singolo) per un sottoinsieme di istruzioni del MIPS
  - [PH] parr. 4.1, 4.3
- Nella prossima lezione aggiungeremo all'Unità di Elaborazione Dati costruita, la relativa Unità di Controllo
  - In realtà, una Unità di Controllo della ALU e una Unità di Controllo Principale
- In seguito studieremo una seconda implementazione, più realistica, del processore MIPS, basata su pipeline