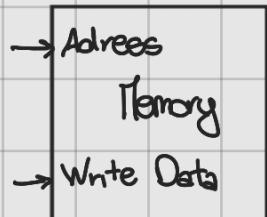


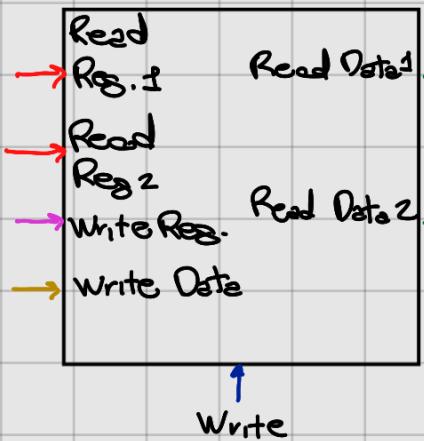
# DATA PATH TEORIA

## 2.3. Componenti Data path

Memory: Contiene istruzioni e dati



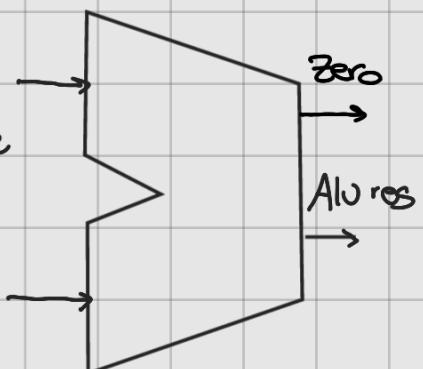
Register File: contiene i dati utilizzati durante l'esecuzione delle istruzioni.



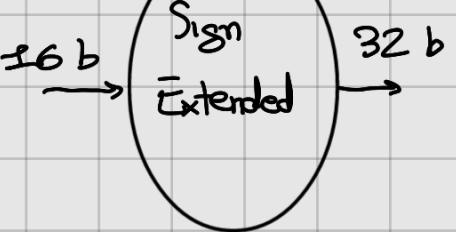
- Legge due registri
- Restituisce in output due registri
- Segnale per decidere se scrivere o meno
- Segnale che contiene il dato da scrivere
- Registro che contiene dove scrivere

Alu

Quali op.	Cosa fa?
tutte	Incrementa PC + 4 per andare all'istruzione successiva.
beq	Calcola valore della branch
R-type	Esegue operazione
lw, sw	Calcola indirizzo a cui andare



Sign Extended: Estensione di segno da 30 bit a 32 bit



Shift Left 2 : shift a sinistra di due zeri (moltiplica per 4)



PC: contiene i 32 bit dell'istruzione da eseguire

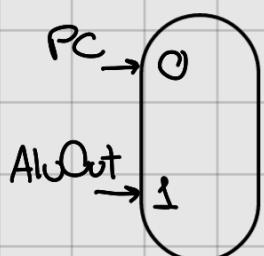
IR: contiene la codifica dell'istruzione

MDR: contiene il dato letto da memoria prima della scrittura nel reg. destinazione

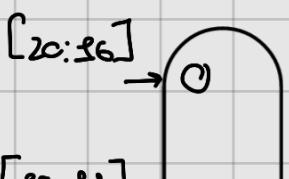
A, B: contengono gli output del Register File

ALUOut: contiene l'output della Alu

## 1.2 Multiplexor nel Datapath

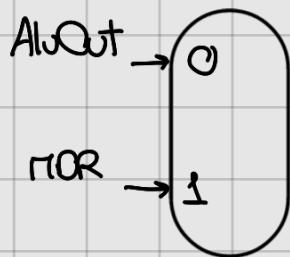


TRA PC È MEMORY. SCEGLIE TRA:  
 $0 = \text{PC} (\text{legge istr.})$     $1 = \text{AluOut} (lw)$

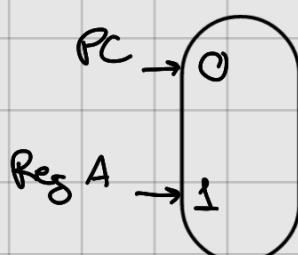


TRA IR È REGISTER FILE. SCEGLIE TRA:  
 I-Type(0) , R-Type(1)

LSS:  $\Rightarrow$  1



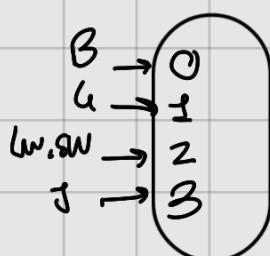
Tra MDR è REGISTER FILE: segue tra  
AluOut (R-type) = 0, MDR (load) = 1



Primo cf.

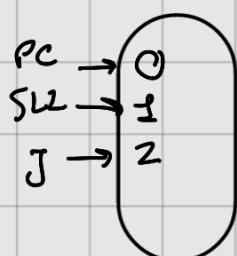
Tra REGISTER FILE è AW. SEGUENTRA

PRIMO OLFANDO: Tra PC(+4b) e Reg. A (R-type) = 1



SECONDO CF. TRA;

- B (R-Type) = 0      • SIGNEXT. (LW, SW) = 2
- U (PC) = 1      • SLZ (J) = 3



AGGIORNAMENTO PC:

- AW (PC + 4) = 0
- AW Out (SLZ) = 1
- J target address [25:0] = 2

### §.3. Cicli necessari:

R-type, SW = 4

lw = 5

beq, j = 3

### §.4. Chi utilizza un componente

Scrittura di filo 81 - 1

3. JUILLIA reg. File: R-Type, lw

2. Reg A : tutte tranne jump

3. Reg B : R-type + beq + sw (se non specifica nella "Alu")

4. due o + somme: tutte

5. due somme: R-type tranne add, beq, j

6. tre somme: add, lw, sw

7. 3 sottrazione / 2 somme o 3 sottrazione: sub, beq NO SLT

### 1.5. Differenza multiciclo e singolo ciclo

Nell'implementazione a multiciclo spezziamo l'istruzione in fasi, dove ciascuna avviene in un suo ciclo di clock.

Così facendo i cicli sono + corti e possiamo ri-usare le unità più volte per istruzione, riducendo l'hardware necessario.

I tempi sono ottimizzati perché nel singolo ciclo tutte le operazioni si adattano a quella più lunga (lw) mentre qui ognuna "ha i suoi tempi" lw 5 cicli,

Rtype sw 4 cicli

beq j 3 cicli

### 1.6. Control Unit

La Control Unit riceve in ingresso l'opcode dal Instruction Register e attraverso essa capisce quali segnali accendere / spegnere.

Se abbiamo una R-Type non ci basta le CU, ma necessitiamo dell' ALU Control che riceve il funz ed interpreta che operazione dobbidmo eseguire.

## Schema

## Data path

