Realizzazione di un Datapath

Leonardo Bizzoni

May 2, 2023

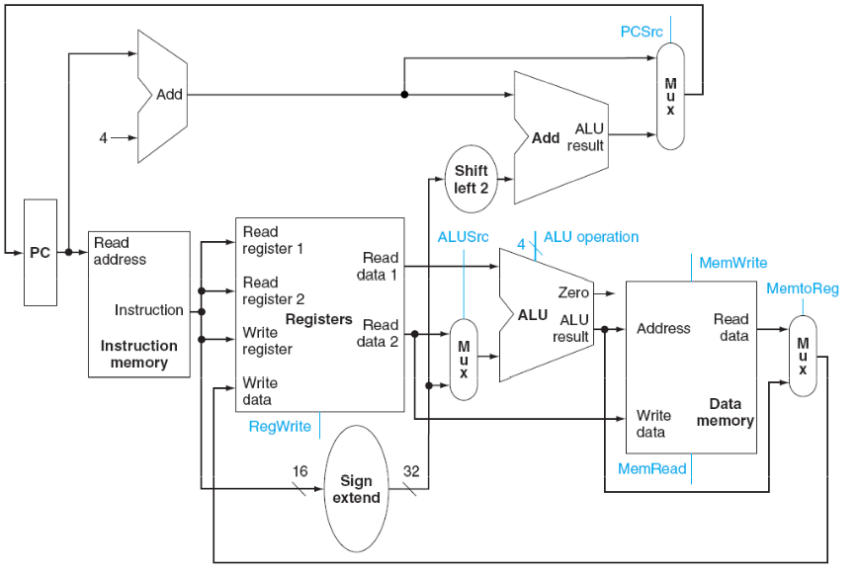
Per realizzare un datapath è necessario come prima cosa stabilire il set di istruzioni da implementare, i vari componenti utili per eseguire tali istruzioni e una metodologia di clocking per interagire con gli elementi di memorizzazione.

Considerando un processore MIPS che esegue solo le istruzioni:

- **artimetico-logiche**: add, sub, and, or, slt
- di **iterazione con la memoria**: lw, sw
- di **salto**: beq, j

Avremo bisogno di:

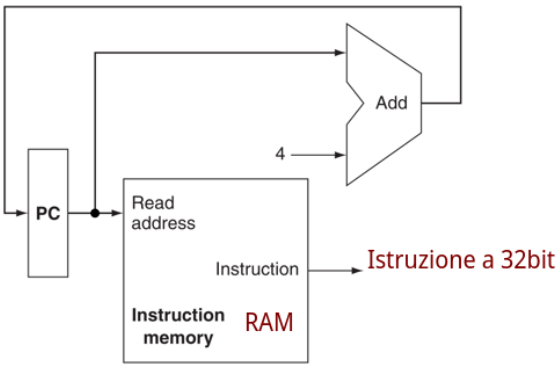
- una unità di memoria per memorizzare le istruzioni di un programma (**instruction memory**)
- una unità di memoria per memorizzare i dati di un programma (**data memory**)
- un registro per contenere l'indirizzo di memoria dell'istruzione corrente (**Program Counter**)
- Un **adder** per incrementare il valore del PC
- Una **ALU** per effettuare le operazioni aritmetico-logiche
- Un **Register File** formato da 32 registri utilizzabili da un programma



1 Eseguire un'istruzione

1.1 Fetch - lettura di un'istruzione

Per poter eseguire un'istruzione è necessario prima leggere dalla memoria specificando un indirizzo specificato dal program counter. Una volta letta l'istruzione, **indipendentemente** dal tipo di essa, il program counter viene incrementanto(+4) in modo tale da puntare alla prossima operazione.



1.2 Decode - interpretazione di un'istruzione

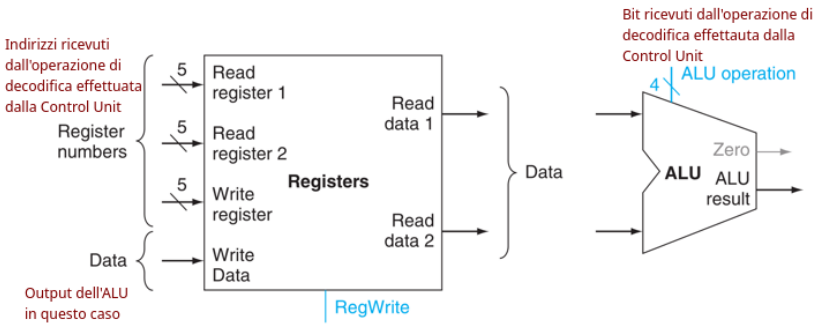
Una volta letta l'istruzione il processore MIPS legge i vari campi dell'istruzione e ne identifica il tipo tramite OPCODE e FUNC.

1.3 Execute

1.3.1 Operazioni aritmetico-logiche

Tutte le istruzioni aritmetico-logiche:

- leggono da esattamente 2 registri
- utilizzano l'ALU per effettuare una specifica operazione con i valori dei 2 registri
- scrivono il risultato in un terzo registro.



1.3.2 Load e Store - *lw/sw $t1, offset($base)*

Per entrambe le istruzioni è necessario calcolare l'indirizzo da cui dobbiamo leggere o scrivere. L'indirizzo viene calcolato tramite un'operaizone di somma tra il registro base e l'offset a 16bit.

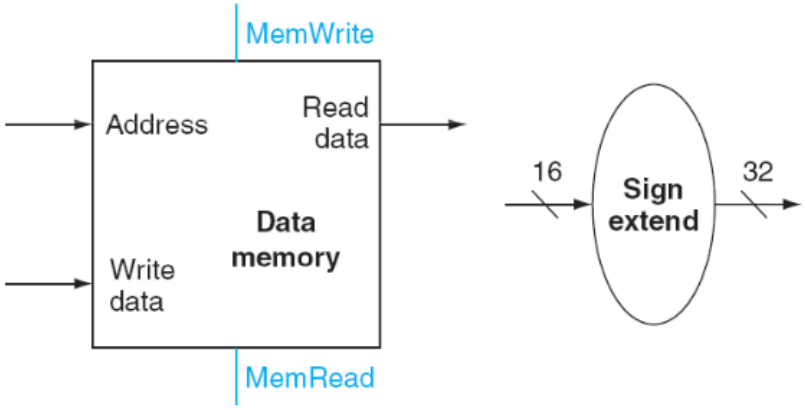
Dato che l'offset può essere negativo esso viene rappresentato utilizzando il CA2. L'operazione di somma deve essere fatta tra 2 numeri con lo stesso numero di bit quindi si ha la necessità di **estendere il bit di segno** dell'offset per ottenere un valore a 32bit.

L'indirizzo appena calcolato andrà in input alla memoria dati.

Per l'operazione di **store** oltre all'indirizzo verrà dato in input:

- il valore contenuto nel registero indicato dall'istruzione
- un segnale asserted per indicare la possibilità di scrivere in quell'indirizzo

Per l'operazione di **load** il valore contenuto nella memoria dati all'indirizzo fornito verrà scritto all'interno del register file al registro indicato dall'istruzione.

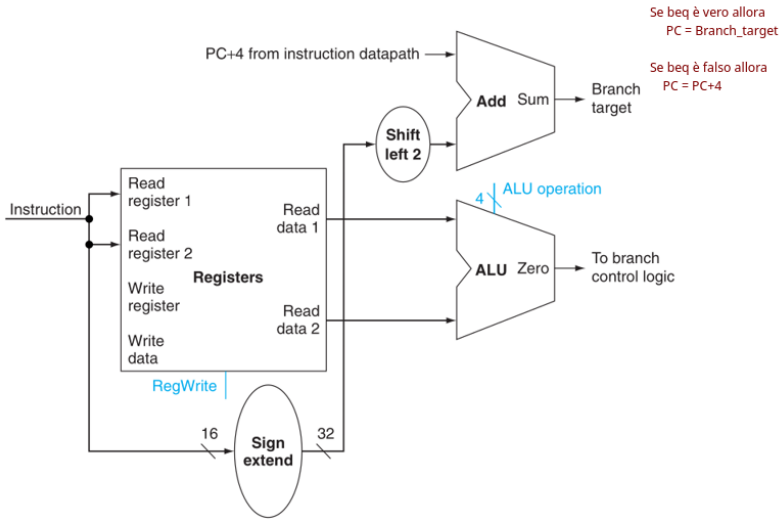


1.3.3 BEQ - *beq $t1, $t2, offset*

L'operazione di *beq* riceve in input 2 registri di cui controlla l'uguaglianza e un *offset* a 16bit che verrà sommato al PC in caso di verificata uguaglianza.

Nell'ISA è specificato che il calcolo dell'indirizzo di destinazione della branch viene effettuato:

- con indirizzo di base quello della **prossima** istruzione ($PC+4$)
- l'offset deve riguardare le word e non i bit singoli ($offset«2$)



1.3.4 J - *j offset*

L'operazione di jump incondizionato semplicemente sostituisce i 28bit meno significativi del PC con i 26bit forniti dall'offset (*offset «2*).

2 Metodologia di clocking

Single-cycle	Multi-cycle
Un'istruzione viene eseguita in un ciclo di clock	Un'istruzione può venire eseguita in più cicli di clock
Ciclo lungo di lunghezza fissa	Ciclo corto di lunghezza fissa

Utilizzando un metodologia di clocking single-cycle si ha uno spreco di tempo in quanto anche le istruzioni che richiedono meno tempo devono attendere la fine del ciclo per permettere ad istruzioni più lunghe di terminare nello stesso periodo di tempo.

Questo non è un problema per il clocking multi-cycle in quanto il periodo di clock viene impostato in base all'istruzione che richiede meno tempo e le istruzioni più lunghe possono utilizzare più cicli per terminare.