Architettura degli Elaboratori I - B

Le Microarchitetture

Introduzione

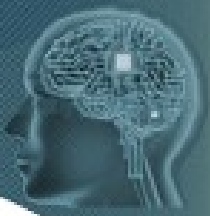
Daniel Riccio/Alberto Aloisio
Università di Napoli, Federico II

8 marzo 2018

Rif. Capitolo 7

Digital Design and Computer Architecture-ARM, Harris-Harris, Edition-Morgan Kaufmann.

Sommario

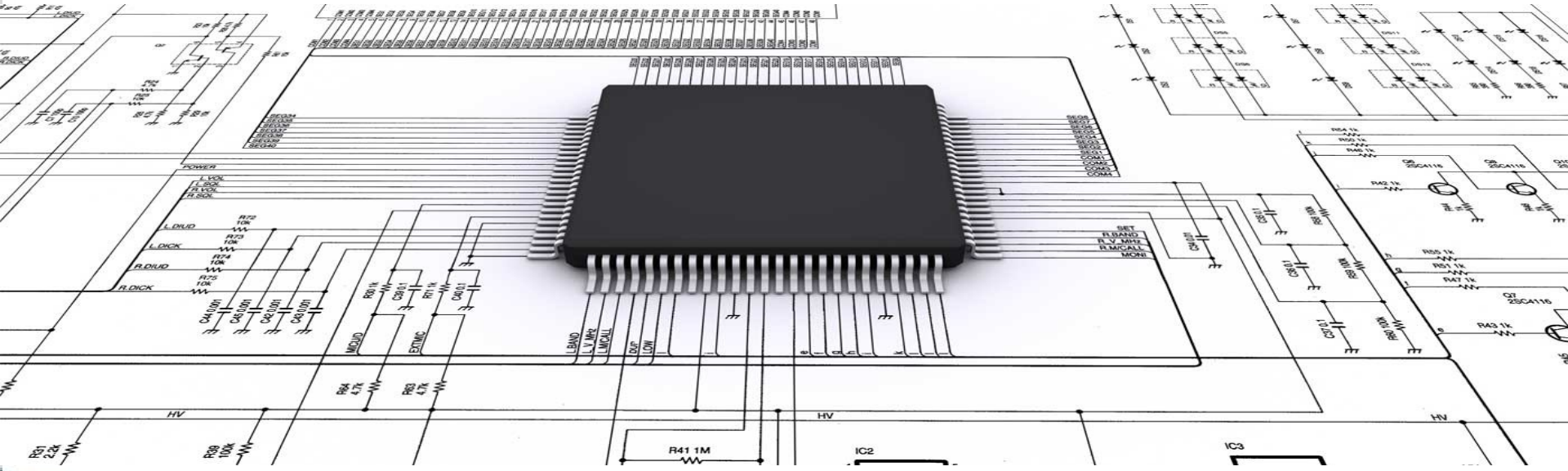


- Introduzione alle microarchitetture;
- Il microprocessore ARM;
- gli elementi di stato di un processore ARM.

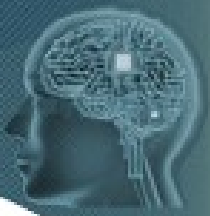
Introduzione

La progettazione di un processore general-purpose richiede la considerazione attenta di molti aspetti. Data la natura complessa dei processori reali, l'**astrazione** è un requisito fondamentale per una piena comprensione dei principali aspetti di un processore.

La nostra attenzione si concentrerà principalmente sulla progettazione **hardware** di un processore e sulle caratteristiche del suo **set di istruzioni**.



L’evoluzione



Tuttavia, non sempre la tecnologia è stata la principale fonte del cambiamento. In alcuni casi, sono stati fattori esterni ad influenzare la direzione in cui la tecnologia si è evoluta.

- **Computer architecture**: essa descrive il computer dal punto di vista dell’utente, influenzando il set di istruzioni, i registri visibili, le strutture di gestione delle tabelle delle pagine di memoria, la gestione delle eccezioni.

- **Computer organization**: descrive meccanismi trasparenti per l’utente, quali la struttura della pipeline, le memorie cache, il translation look-aside buffer.

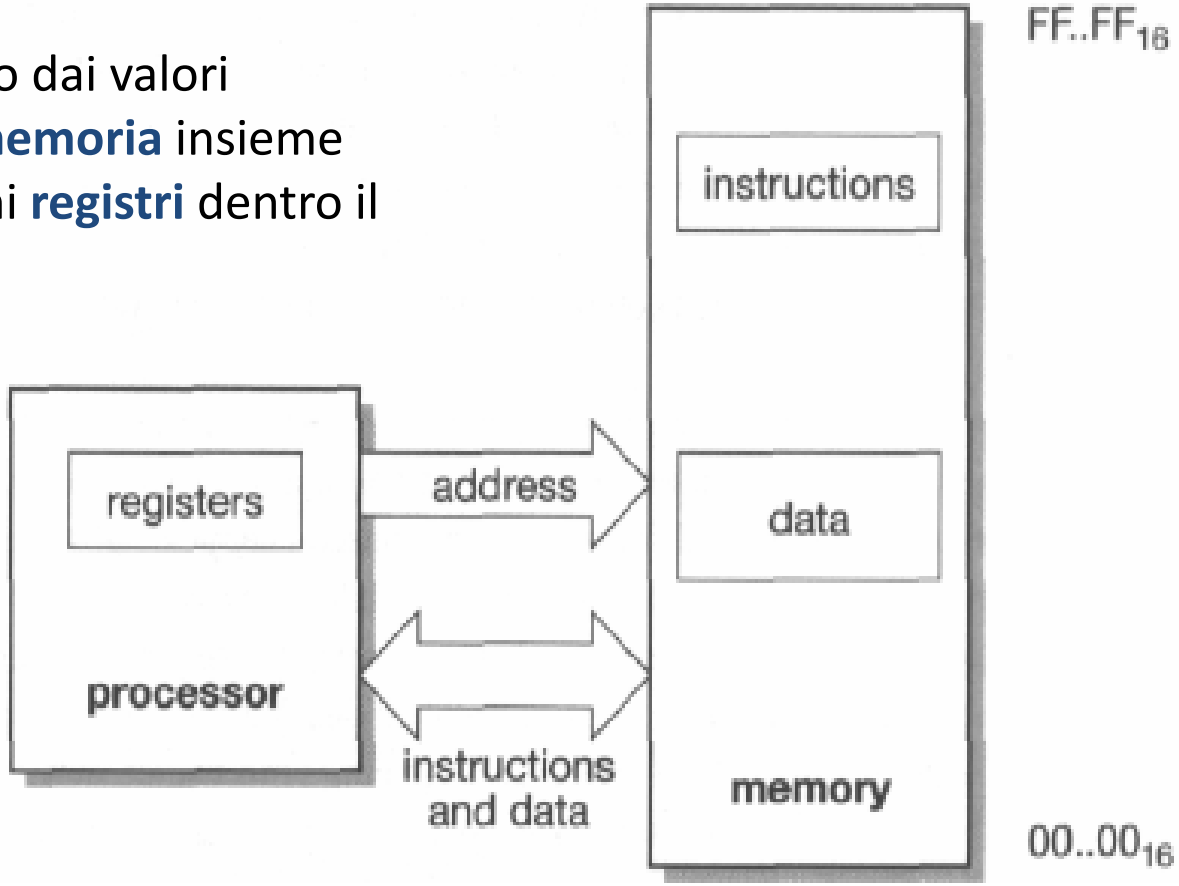
Fra i progressi in questi aspetti della progettazione dei computer, l'introduzione della **memoria virtuale** nei primi anni sessanta, delle **memorie cache**, del **pipelining** e così via, sono state tutte delle pietre miliari per l'evoluzione dei computer.

Cos'è un processore

Un processore general-purpose è un **automa** a stati finiti, che esegue **istruzioni** rilocate in una memoria.

Lo **stato** del sistema è definito dai valori contenuti nelle locazioni di **memoria** insieme con i valori contenuti in alcuni **registri** dentro il processore stesso.

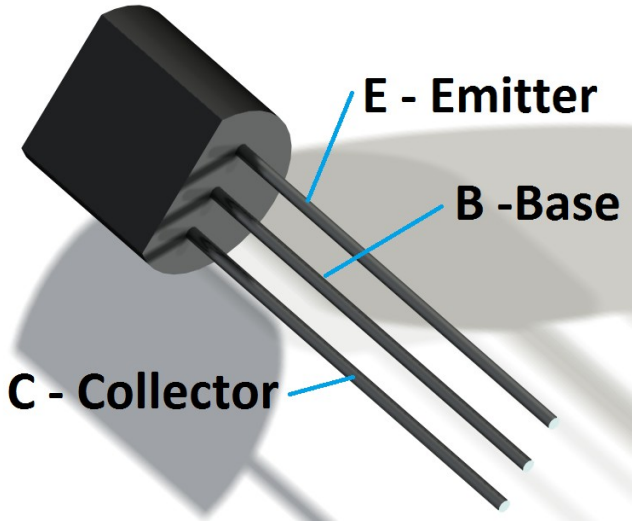
Ogni **istruzione** definisce in che modo lo stato deve cambiare e quale istruzione deve essere eseguita successivamente.

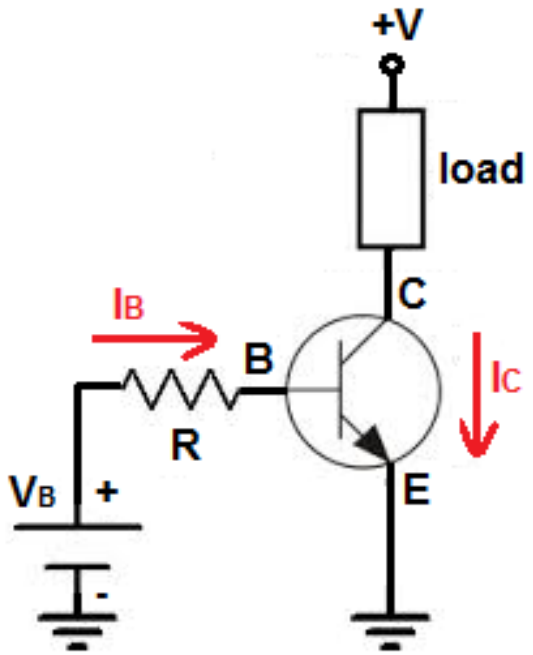


Come è fatto …

I computer sono una complessa sintesi di apparecchiature che operano a velocità molto elevate.

Un moderno microprocessore è costituito da diversi milioni di transistor, ciascuno dei quali può cambiare il suo stato centinaia di milioni di volte in un secondo.



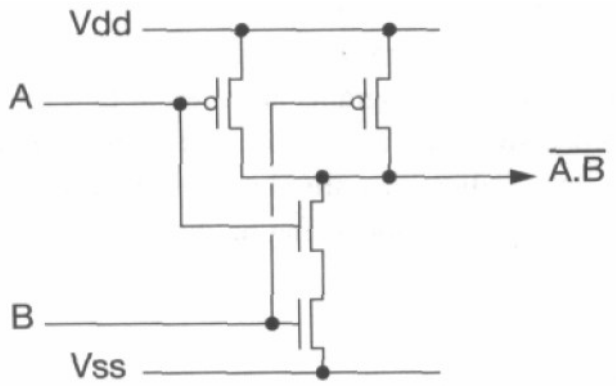


In an NPN transistor, positive voltage is given to the collector terminal and current flows from the collector to the emitter, given there is sufficient base current

Dal transistor alla porta logica

Le equazioni che descrivono il comportamento di un transistor sono piuttosto complesse.

Quando un gruppo di transistor è collegato insieme in una particolare struttura, come il CMOS (Complementary Metal Oxide Semiconductor) o porta **NAND**, il comportamento del gruppo può essere descritto in modo molto più semplice.



Associando Vdd a **true** e Vss a **false**:

output = —(A^B)

Se ciascuna delle linee di ingresso (A e B) è mantenuta ad una tensione vicina a Vdd o a Vss, l'uscita sarà vicina a Vdd e Vss secondo le seguenti regole:

- Se **A** e **B** sono entrambe vicino a **Vdd**, l'uscita sarà vicina a **Vss**.
- Se **A** o **B** (o entrambi) sono vicine a **Vss**, l'uscita sarà vicina a **Vdd**.

Dalla porta logica alla sua astrazione

Il processo di **astrazione** ha due principali vantaggi

- Permette di **semplificare** notevolmente il processo di progettazione di circuiti con un gran numero di transistori.

- Elimina la necessità di sapere che la porta è costruita da transistor.

Un circuito logico dovrebbe avere lo stesso comportamento logico, a prescindere da come sono realizzate le porte logiche.

La tecnologia di implementazione influisce sulle prestazioni del circuito ma non dovrebbe avere alcun effetto sulla sua funzione.

Dalla porta logica alla sua astrazione

L'astrazione di una porta logica si fonda su due elementi:

- **simbolo logico**: che rappresenta la funzione di una porta in uno schema circuitale (ad esempio la porta NAND).

- **tabella di verità**: descrive la funzione logica della porta ed è tutto ciò che il progettista deve sapere su di essa per la maggior parte degli scopi.

Logic symbol

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

Livelli di astrazione

Il processo di astrazione è ripetuto a **diversi livelli** in computer science ed è assolutamente fondamentale per la gestione della **complessità**.

Il processo di raggruppare componenti ad un livello per descrivere il loro comportamento comune, nascondendo i dettagli non necessari al livello successivo, permette di scalare gli ordini di complessità in pochi passaggi:

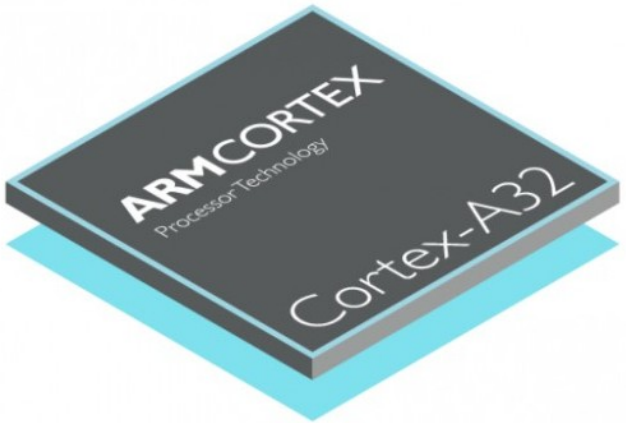
1. transistor;
2. Porte logiche, celle di memoria, circuiti specifici;
3. Addizionatori single-bit, multiplexer, decoder, flip-flop;
4. Addizionatori word-wide, multiplexer, decoder, registri bus;
5. ALUs (Arithmetic-Logic Units), registri a scorrimento, banchi di registri, blocchi di memoria;
6. processore, cache e gestione della memoria;
7. processori, memorie periferiche, cache, memory management unit;
8. integrated system chip;
9. Circuiti stampati;
10. Smartphone, Tablet, PC, engine controller.

Le microarchitetture

Una Microarchitettura rappresenta il collegamento tra la logica e l'architettura.

Essa definisce la disposizione specifica di registri, ALU, macchine a stati finiti (MSF), le memorie e altri blocchi logici necessari per implementare una architettura.

Un particolare architettura, come l'ARM, può avere molti microarchitetture diversi, ciascuno con diversi livelli di prestazioni, costi e complessità.



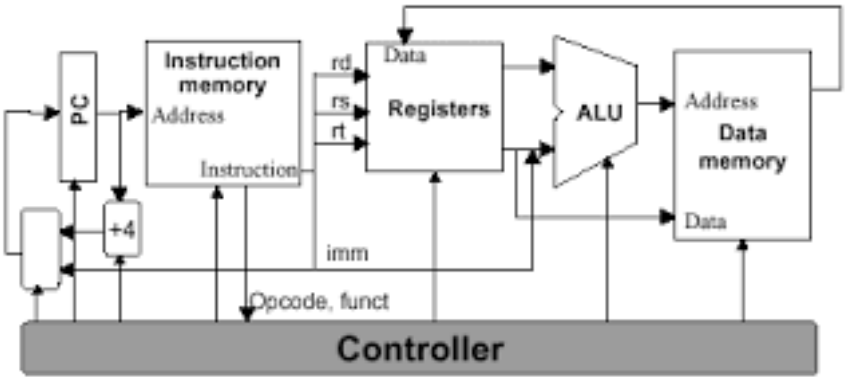
Le microarchitetture

L'architettura di un computer è definita da due elementi:

- Il suo set di istruzioni
- Lo stato architettonico.

Lo stato architettonico per il processore ARM consiste di 16 registri a 32 bit e del registro di stato.

Mnemonic	Instruction	Action
ADC	Add with carry	Rd:=Rn+Op2+Carry
ADD	Add	Rd:=Rn+Op2
AND	AND	Rd:=Rn AND Op2
B	Branch	R15:=address
BIC	Bit Clear	Rd:=Rn AND NOT Op2
BL	Branch with Link	R14:=R15 R15:=address
BX	Branch and Exchange	R15:=Rn T bit:=Rn[0]
CDP	Coprocessor Data Processing	(Coprocessor-specific)
CMN	Compare Negative	CPSR flags:=Rn+Op2
CMP	Compare	CPSR flags:=Rn-Op2



Qualsiasi microarchitettura ARM deve contenere almeno questi elementi.

In base allo stato corrente, il processore esegue una particolare istruzione con uno specifico insieme di dati al fine di produrre un nuovo stato.

Le variazioni di stato

I registri, la memoria dati e la memoria istruzioni operano mediante logica combinatoria.

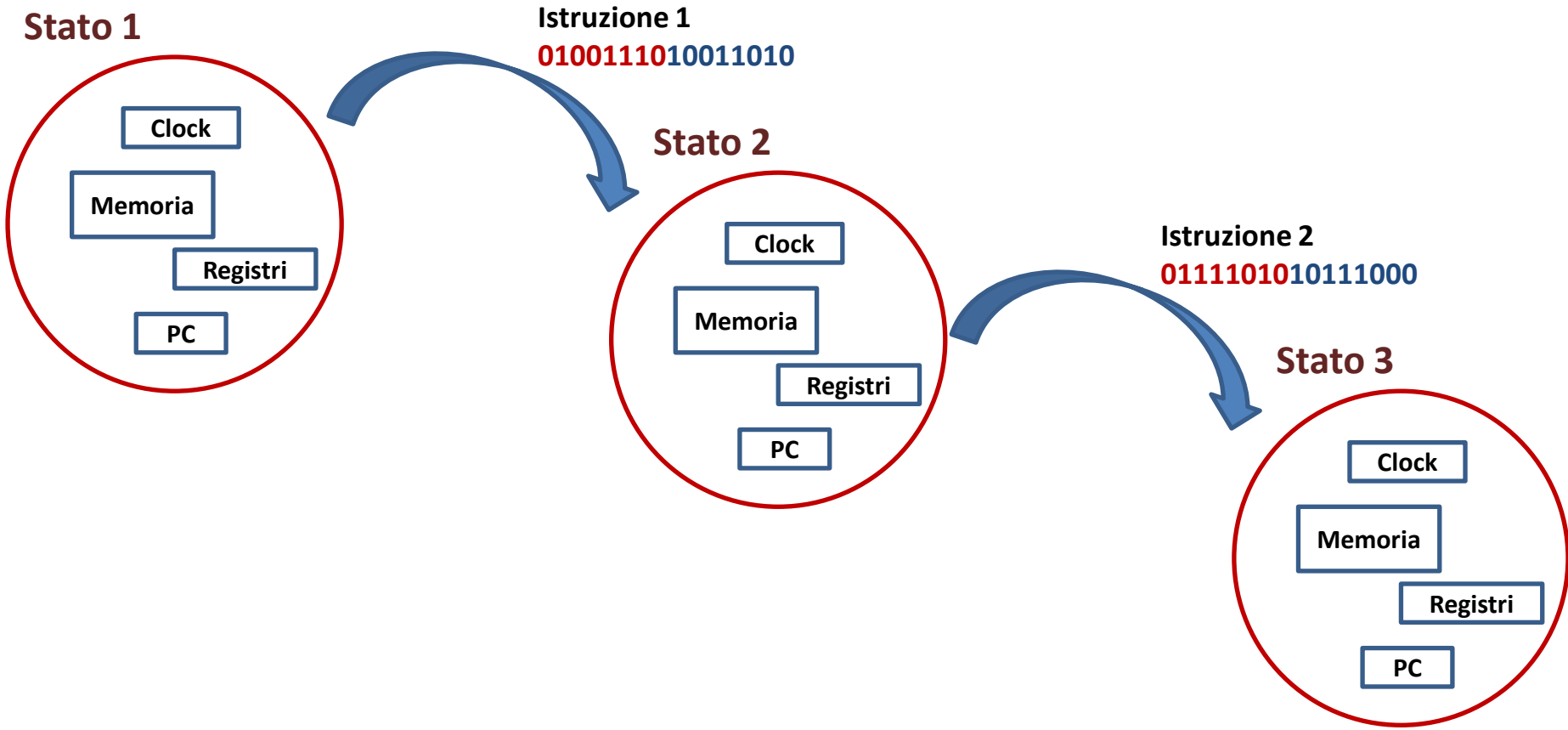
Tutti i componenti effettuano la scrittura sul fronte alto del clock, cosicché lo stato del sistema cambia solo su un fronte del clock.

Gli indirizzi, i dati ed il segnale di write enable devono essere impostati prima del fronte del clock e mantenuti immutati per un tempo superiore al ritardo di propagazione.

Sia gli elementi di stato, che il microprocessore sono costituiti da logica combinatoria e da componenti sincronizzate dal clock. L'intero sistema è, quindi, sincrono e può essere visto come una complessa macchina a stati finiti o come l'insieme di macchine a stati finiti semplici, che interagiscono fra loro.

Le microarchitetture

Una microarchitettura può essere vista come un automa.



Microarchitetture a ciclo singolo

Saranno prese in considerazione tre diverse microarchitetture ARM, le quali differiscono fra loro principalmente per il modo in cui gli elementi di stato sono interconnessi.

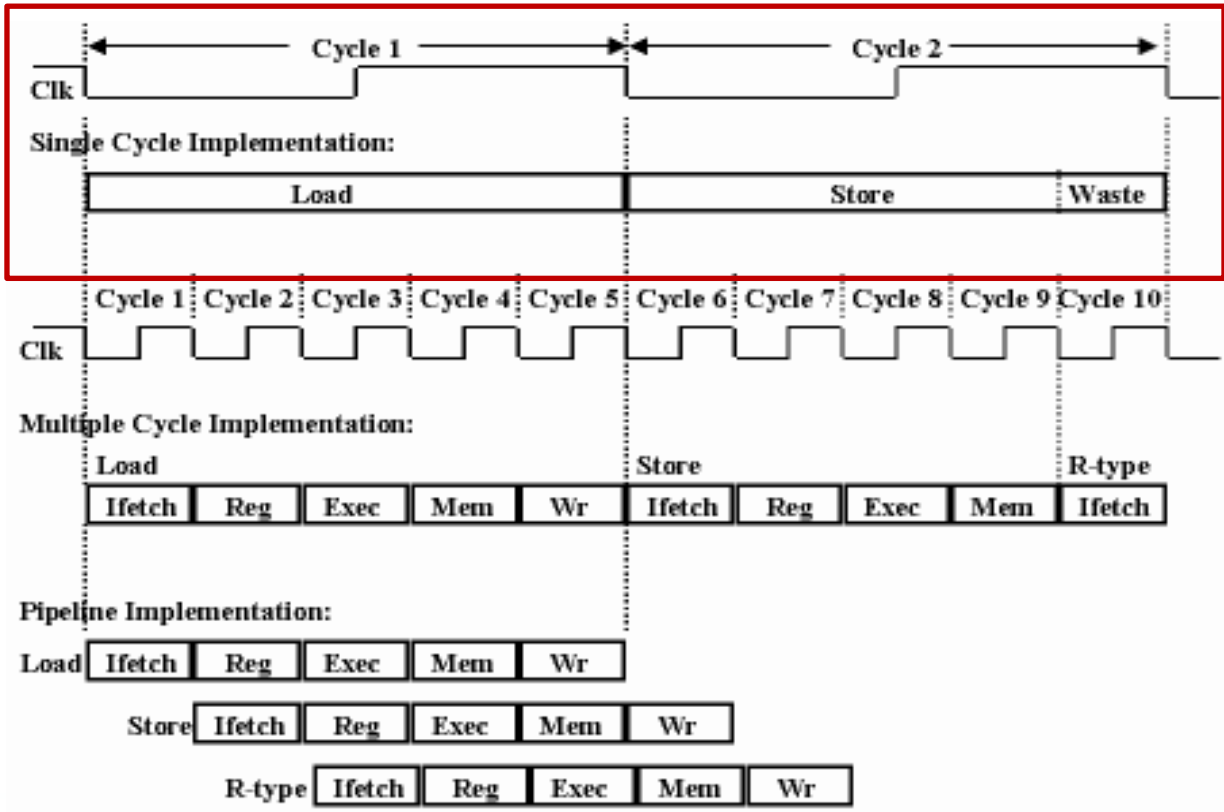
Ciclo singolo: l'intera istruzione è eseguita in un singolo ciclo.

Vantaggi:

- semplice da comprendere;
- unità di controllo molto semplice;
- non richiede stati non architetturali.

Svantaggi:

- tempo pari a quello dell'istruzione più lenta;
- memoria dati e memoria istruzioni separate;



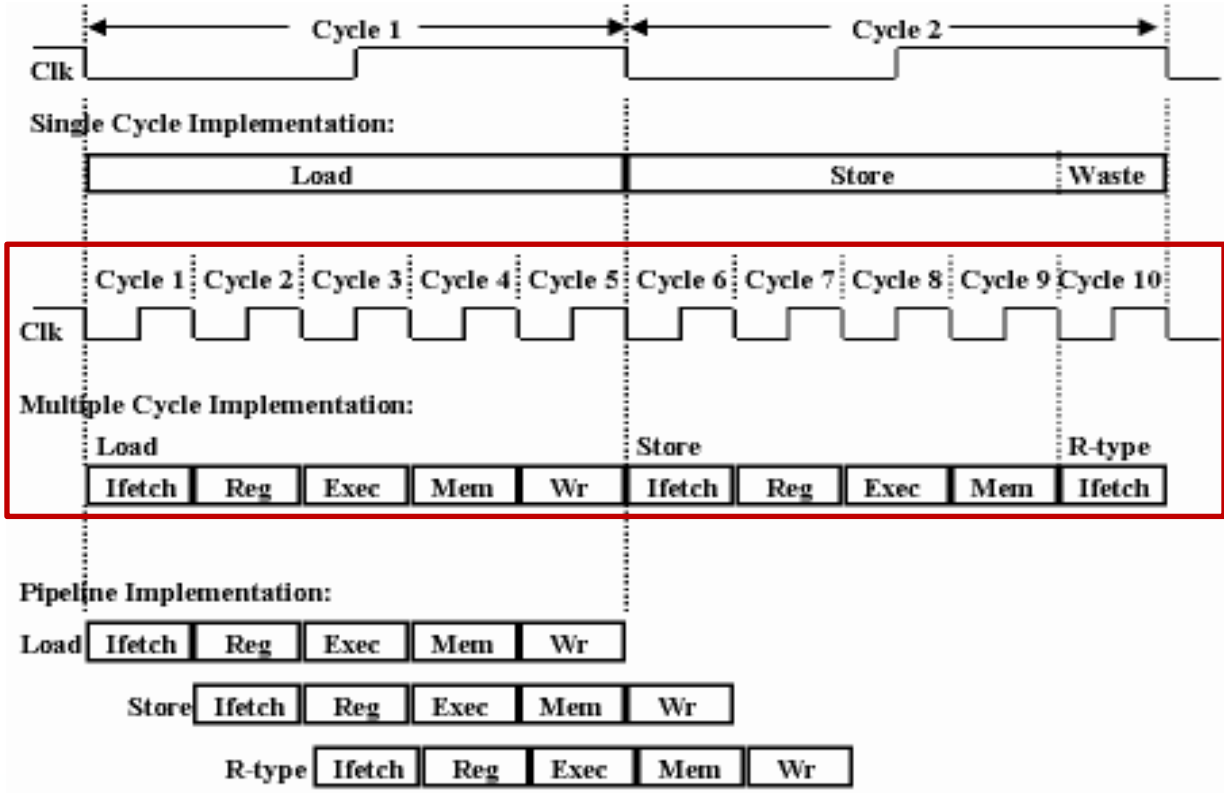
Microarchitetture a ciclo multiplo

Saranno prese in considerazione tre diverse microarchitetture ARM, le quali differiscono fra loro principalmente per il modo in cui gli elementi di stato sono interconnessi.

Ciclo multiplo: esegue una istruzione in più cicli brevi.

Vantaggi:
- riuso dei componenti;
- durata variabile delle istruzioni;
- non richiede la separazione delle memorie.

Svantaggi:
- richiede stati non architetturali;
- esegue una istruzione per volta.



Microarchitetture con pipeline

Saranno prese in considerazione tre diverse microarchitetture ARM, le quali differiscono fra loro principalmente per il modo in cui gli elementi di stato sono interconnessi.

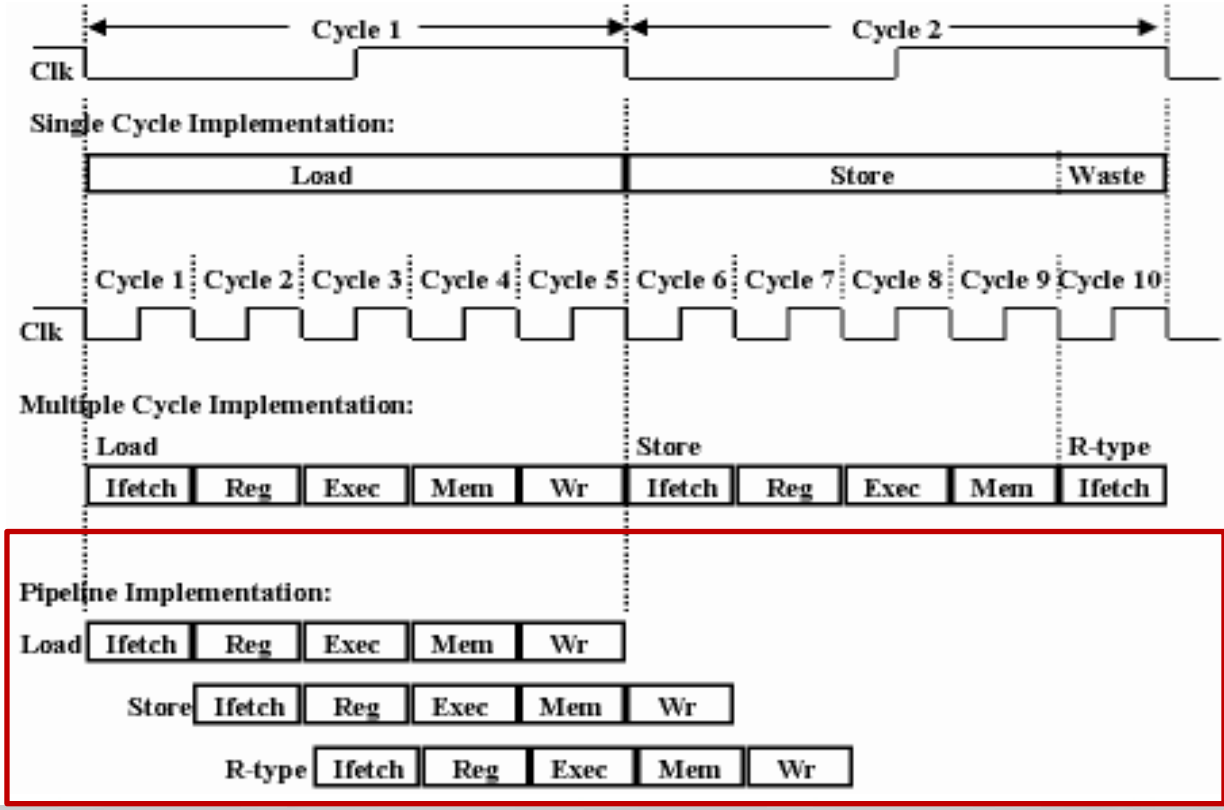
Pipelining: si applica al processore a ciclo singolo e ne migliora le performance.

Vantaggi:

- esegue più istruzioni contemporaneamente;
- si può accedere a dati e registri contemporaneamente.

Svantaggi:

- logica di controllo più complessa.
- richiede registri di pipeline.



La misura delle prestazioni

Uno stesso processore può avere diverse microarchitetture con differenti costi e prestazioni.

Ci sono molti modi per misurare le performance di un processore e la frequenza del clock è solo una di queste.

Una misura più affidabile consiste nel valutare le prestazioni di tempo rispetto all'esecuzione di un insieme fissato di programmi, che prende il nome di benchmark.

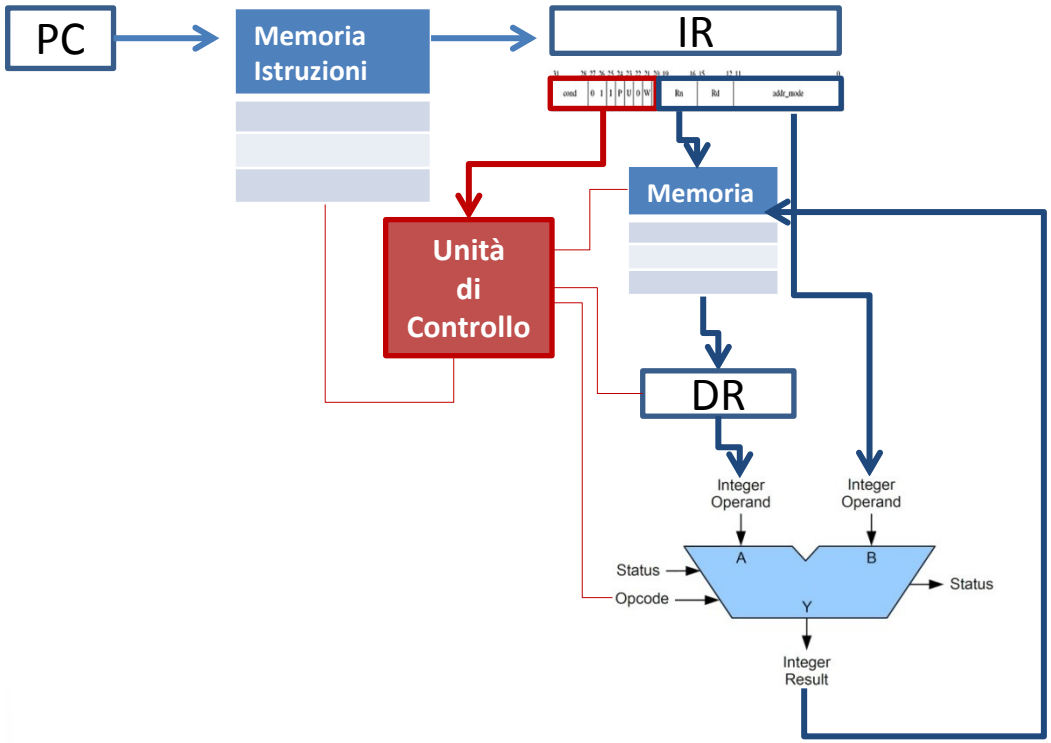
Il tempo di esecuzione è calcolato come:

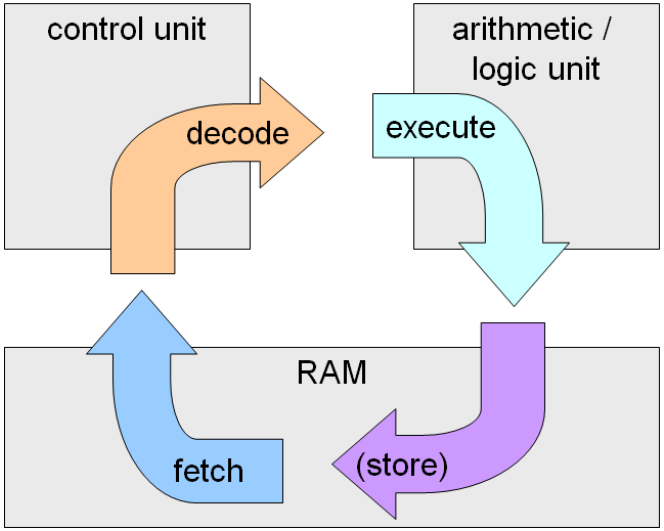
$$Tempo\ di\ esecuzione = (\#istruzioni)\left(\frac{cicli}{istruzione}\right)\left(\frac{secondi}{ciclo}\right)$$

Lo scopo del progettista consiste nel produrre una architettura che, pur rispettando i vincoli di costo, minimizzi il tempo di esecuzione e riduca il consumo di energia.

Le microarchitetture

Una istruzione ha un ciclo di vita che consta di quattro fasi principali

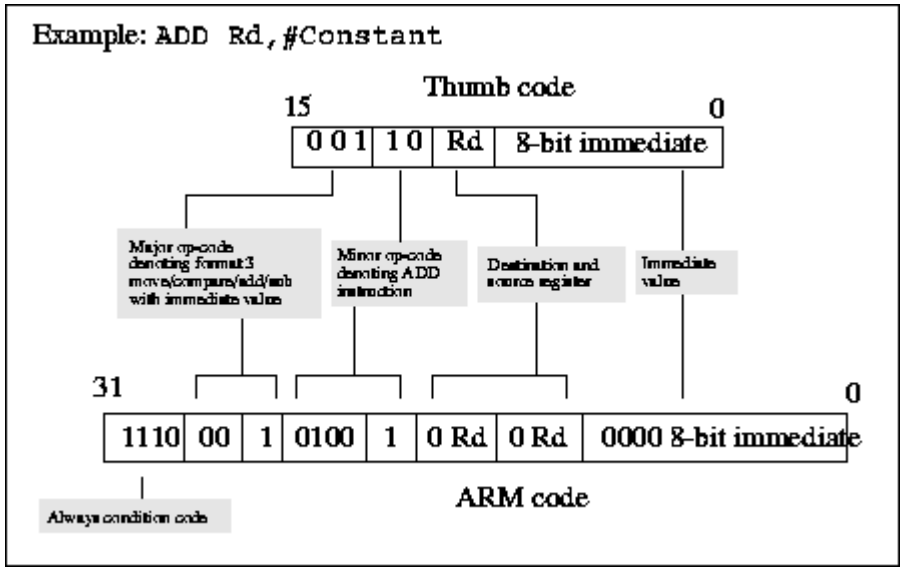




Le microarchitetture

Al fine di facilitare la comprensione dell'architettura di un processore ARM, considereremo un limitato set di istruzioni:

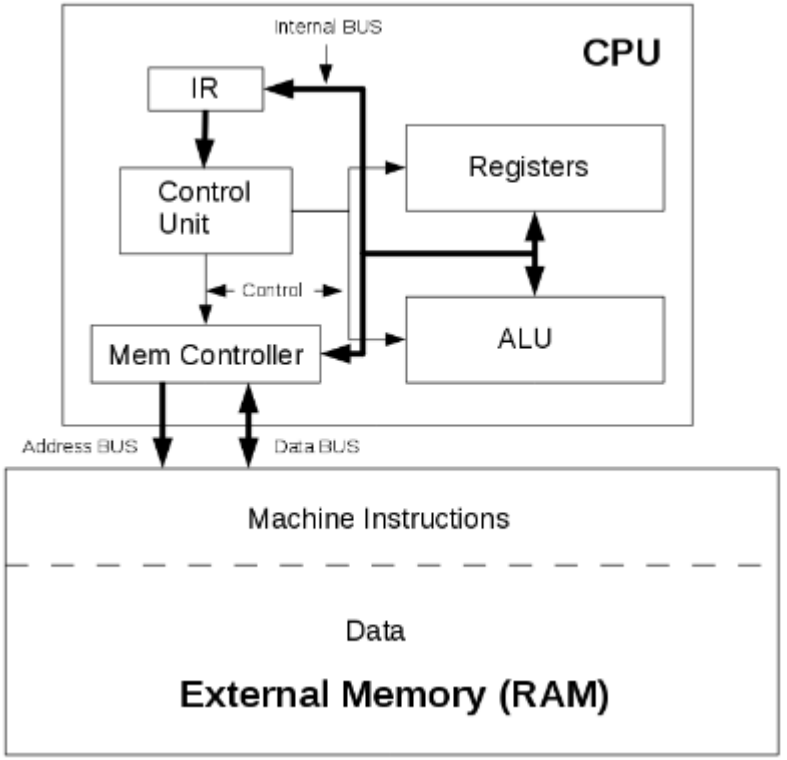
- le istruzioni di elaborazione dati: **ADD**, **SUB**, **AND**, **ORR** (con registro e modalità di indirizzamento diretto e senza shift).
- le istruzioni di Memoria: **LDR**, **STR** (diretto con offset positivo)
- le istruzioni di salto: **B**



Progettazione

Da un punto di vista della progettazione, dividiamo la nostra microarchitettura in due componenti che interagiscono fra loro:

- il **datapath** – opera su parole dati e contiene strutture quali le memorie, i registri, l'ALU, e i multiplexer. Considereremo un datapath a 32 bit.
- l'**unità di controllo** – riceve l'istruzione corrente dal datapath e dice al datapath come eseguire tale istruzione.



Procederemo in modo incrementale:

- Elementi di stato (program counter, registri, e registro di stato).
- Logica combinatoria fra gli elementi di stato;
- Gestione della memoria;

Progettazione

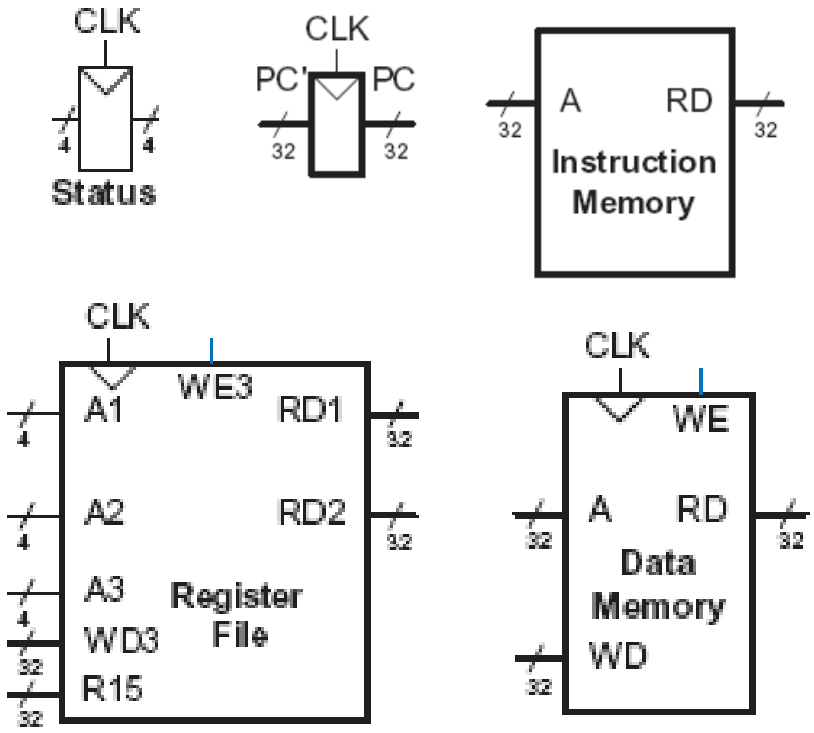
Il miglior modo di procedere nel processo di progettazione consiste nel considerare prima gli elementi di stato (program counter, registri, registro di stato) e aggiungere successivamente la logica combinatoria.

Gli elementi di stato sono cinque:

- il program counter
- i registri (register file);
- il registro di stato (status register);
- la memoria istruzioni (instruction memory);
- la memoria dati (data memory).

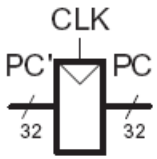
La memoria per le istruzioni e quella per i dati sono separate per il solo scopo di facilitare la comprensione.

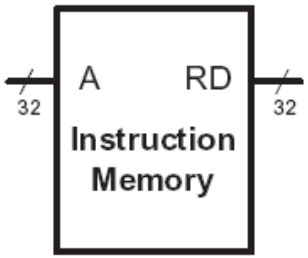
Lo spessore delle linee indica il numero di pin di un bus (spesso-32, medio-16, sottile-1).

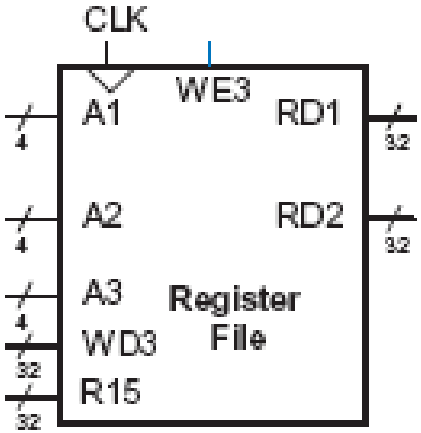


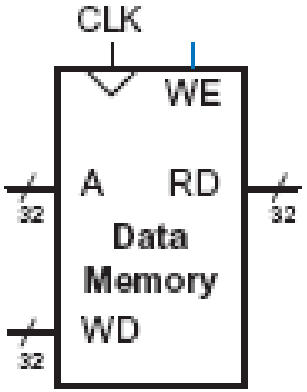
Il datapath

Il miglior modo di procedere nel processo di progettazione consiste nel considerare prima gli elementi di stato (program counter, registri, registro di stato) e aggiungere successivamente la logica combinatoria.



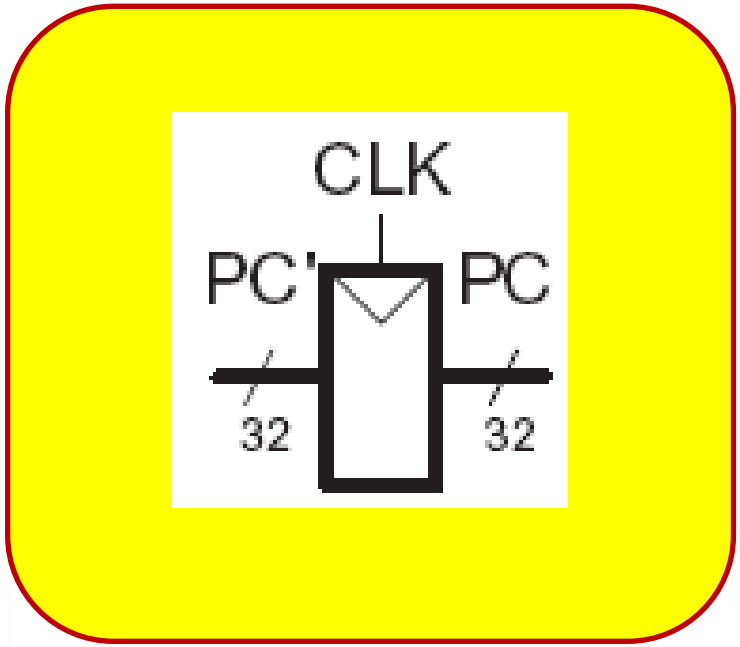


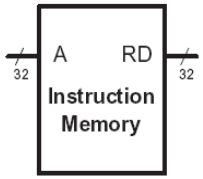


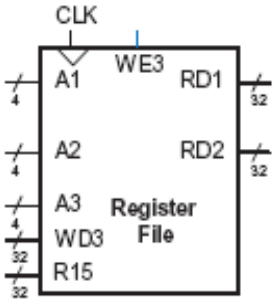


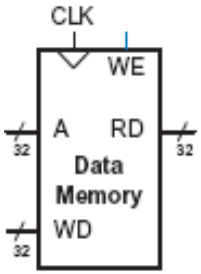
Il datapath

Program Counter: sebbene il PC sia logicamente parte del file registro, esso viene letto e scritto ad ogni ciclo indipendentemente dagli altri registri ed è quindi implementato come un registro autonomo 32-bit. La sua uscita, PC, indica l'istruzione corrente. Il suo ingresso, PC', indica l'indirizzo della successiva istruzione.



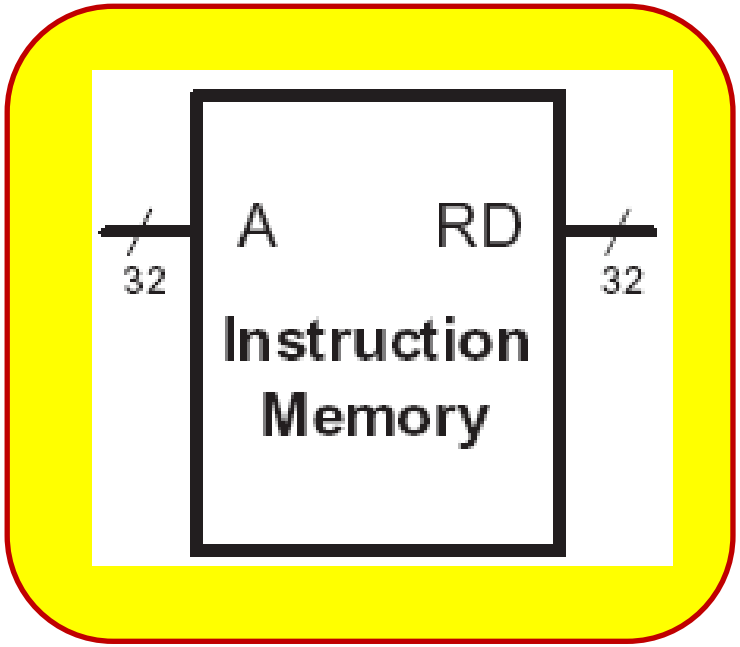


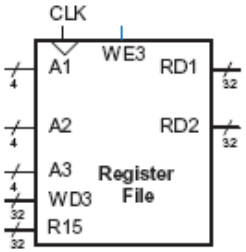


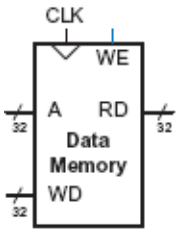


Caratteristiche dei singoli componenti

Instruction Memory: essa ha una singola porta di sola lettura (semplificazione). Prende in input un indirizzo **A** (32 bit) e legge l'istruzione nel registro **RD**.

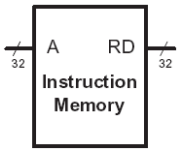


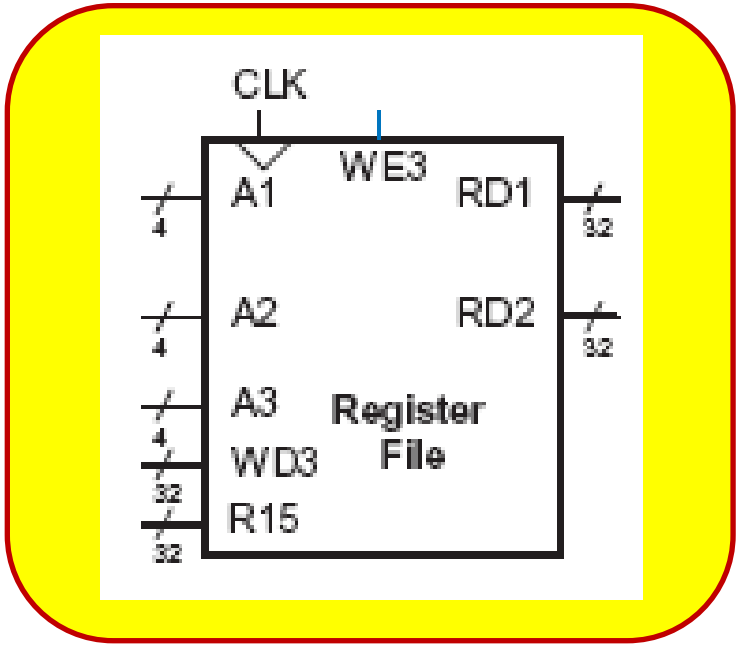


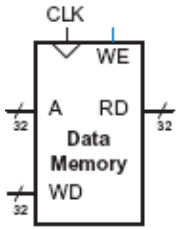


Caratteristiche dei singoli componenti

File Register: consiste di 15 registri di 32 bit ciascuno da **R0** a **R14**, oltre ad un input aggiuntivo **R15** proveniente da **PC**.





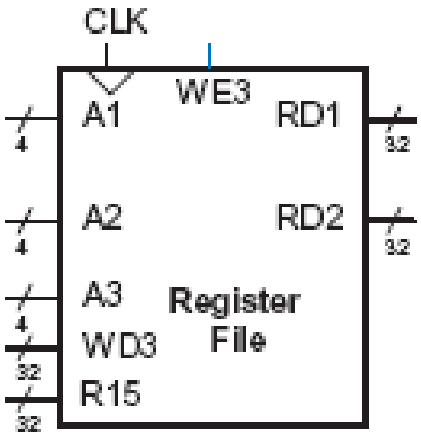


Caratteristiche dei singoli componenti

File Register: consiste di 15 registri di 32 bit ciascuno da **R0** a **R14**, oltre ad un input aggiuntivo **R15** proveniente da **PC**.

Esso ha due porte per la lettura **A1** e **A2** ed una per la scrittura **A3**.

Ciascuna porta di lettura ha un input di 4 bit (2^4=16), che specificano uno dei registri come operando, che viene letto nei registri **RD1** o **RD2**, rispettivamente.



La porta di scrittura ha:

- un indirizzo di 4 bit;
- un elemento di 32 bit **WD3**;
- un segnale di Write Enable (**WE3**);
- il clock.

Se **WE** è 1, il dato è scritto nel registro specificato. Inoltre, viene letto **PC** da **R15**, gli si somma 8 e viene riscritto in **R15**.

Caratteristiche dei singoli componenti

Data Memory: essa ha una singola porta di lettura/scrittura. Quando il segnale **WE** (Write Enable) è attivo, essa scrive i dati nella cella puntata dall'indirizzo **A** durante il fronte alto del clock. Quando il segnale **WE** (Write Enable) è 0, essa legge i dati nella cella puntata dall'indirizzo **A** durante il fronte alto del clock e li pone nel registro **RD**.