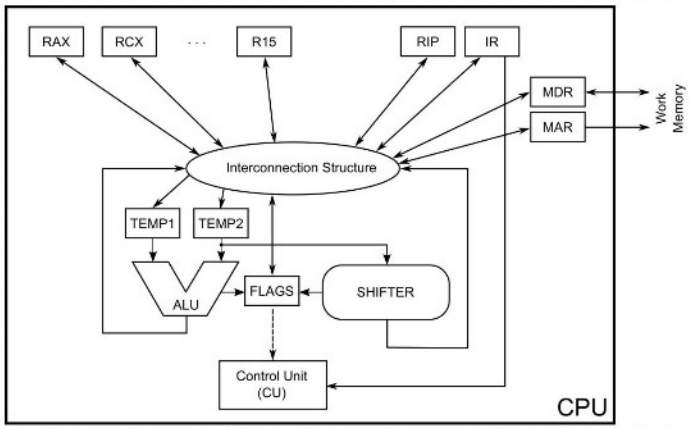
CPU/REGISTRI/CICLO MACCHINA

martedì 8 novembre 2022 14:37

LA CPU

La cpu è una gigantesca RETE combinatoria, con tutto quello che ci siamo detti ora!
La memoria è un altro grande circuito combinatorio!

COME SARÀ COMPOSTO IL NOSTRO PROCESSORE:



Sarà formato da un'UNITÀ di CONTROLLO che GOVERNA TUTTI I CIRCUITI che SONO PRESENTI dentro al NOSTRO PROCESSORE.

DENTRO AVREMO UN INSIEME di REGISTRI: ALCUNI di QUESTI REGISTRI (IN ALTO A SX) SARANNO i REGISTRI VISIBILI AL PROGRAMMATORE, quindi 16 REGISTRI general purpose.

> I registri general purpose (GP) **sono usati per memorizzare i dati durante i calcoli**. Essendo pochi registri non tutti i dati possono essere memorizzati nei registri. Pertanto, occorre spostare in continuazione i dati dalla memoria ai registri.

Poi AVREMO REGISTRI come MDR E MAR, che ci PERMETTONO di FARE UN affaccio sul BUS di SISTEMA per POTER andare a dialogare con la memoria!

ABBIAMO un ALU che CI PERMETTE di fare delle OPERAZIONI logico - ARITMETICHE abbastanza elementari, NIENTE NUMERI IN VIRGOLA MOBILE.
UNO SHIFTER che CI PERMETTERÀ di EFFETTUARE degli SHIFT.
E poi UNA STRUTTURA di INTERCONNESSIONE:
è un INSIEME di FILI che ad esempio PERMETTERÀ alla ALU di PRENDERE dalla MEMORIA INTERNA gli OPERANDI delle OPERAZIONI.

PERMETTERÀ allo SHIFTER di SCRIVERE IL RISULTATO IN OGNUNO di QUEI 16 REGISTRI.

DOBBIAMO capire QUAL'È l'ORGANIZZAZIONE MIGLIORE di QUESTA STRUTTURA di INTERCONNESSIONE PER CONNETTERE TUTTO.

LA SCELTA CHE FAREMO SARÀ UN TRADEOFF TRA COSTI E PRESTAZIONI!

LE NOSTRE ISTRUZIONI SONO CODIFICATE IN BINARIO!
QUESTE PAROLE BINARIE SONO MEMORIZZATE NELLA MEMORIA DI LAVORO;

Quindi l'ESECUZIONE del PROCESSAMENTO delle ISTRUZIONI, dalla MEMORIA di LAVORO, VENGONO PORTATE nel PROCESSORE.

La parola BINARIA viene COPIATA nel PROCESSORE, della MEMORIA, e USERÒ del TEMPO per PROCESSARLA;

PROCESSAMENTO DELLE ISTRUZIONI.

1. FETCH

SERVE a PRELEVARE dalla MEMORIA IL CODICE MACCHINA della PROSSIMA ISTRUZIONE che DOBBIAMO ANDARE AD ESEGUIRE.
QUESTO CODICE MACCHINA DESCRIVE la CODIFICA della NOSTRA ISTRUZIONE!

2. DECODE

Se la NOSTRA PAROLA in CODICE MACCHINA è UNA STRINGA di BIT dobbiamo DECODIFICARE cosa ci

2. DECODE

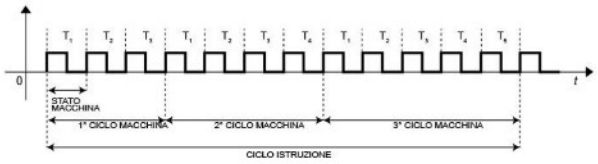
Se la nostra parola in codice macchina è una stringa di bit, dobbiamo decodificare cosa ci sta chiedendo quell'istruzione;

Se nella fase di fetch io leggo come codice macchina "somma" nella fase di decode la CPU deve sapere che voglio andare nella ALU.

3. EXECUTE

L'unità di controllo, guiderà tutti i circuiti che sono disponibili, per eseguire l'istruzione che abbiamo letto.

- *Ciclo istruzione*: intervallo temporale necessario ad eseguire una istruzione nella sua interezza
- *Ciclo macchina*: intervallo temporale necessario ad eseguire una fase (fetch, decode, execute)
  - A seconda del tipo di istruzione, possono essere necessari un numero diverso di cicli macchina
- *Stato macchina*: periodo di tempo necessario per stabilizzare la rete dell'unità di calcolo (corrispondente al clock)



Completare un ciclo istruzione significa arrivare alla fase di fetch dell'istruzione successiva.

PRENDIAMO UN'ISTRUZIONE AD ALTO LIVELLO: $a = a + b$

Memorizza dentro a, la somma tra le variabili a e b. Questo è quello che IO DICO AL PROCESSORE.

1ª SOLA OPERAZIONE: LA SOMMA.
2 VARIABILI COINVOLTE: a e b.

b viene accedita in sola lettura e a viene accedita sia in lettura che in scrittura;

IL PROCESSORE dovrà governare l'HW per dire: "leggi a, leggi b, fai la somma, scrivi a";

a:	15			a:	24
b:	9	a=a+b →		b:	9
	Prima				Dopo

L'esecuzione dovrà portare a questo cambio ↑ di stato.

LA VARIABILE IN CUI SCRIVO IL RISULTATO È NECESSARIAMENTE UNA VARIABILE DI INPUT; QUESTI VALORI STANNO DENTRO A DEI REGISTRI.

DEVO STABILIRE DOVE SONO MEMORIZZATI I VALORI DA SOMMARE, DEVO STABILIRE DOVE SCRIVERE IL RISULTATO; E DECIDERE QUALE OPERAZIONE DEVO ANDARE A SVOLGERE.

GLI OPERANDI SONO REGISTRATI NEI REGISTRI DI USO GENERALE. 1:00:58

QUAL'È IL FORMATO DELLE ISTRUZIONI ASSEMBLY PER DESCRIVERE UN'OPERAZIONE DI QUESTO TIPO?

```
ADDs <sorgente>, <destinazione>
```

Destinazione = destinazione + sorgente

ABBIAMO SEMPRE PARLATO DI CONTESTO.

UNA CERTA INFORMAZIONE VIENE INTERPRETATA IN UN CERTO MODO, A SECONDA DEL CONTESTO CHE GLI DIAMO.

COME DEVONO ESSERE INTERPRETATI QUESTI OPERANDI?

OGNI ISTRUZIONE ALLA FINE AVRÀ UN SUFFISSO, CIOÈ UNA LETTERA E LA LETTERA POSSIBILE SARÀ:

```
s può essere B, W, L, Q—il nome dei registri varia di conseguenza
```

B	W	L	Q
BYTE	WORD	LONGWORD	QUADWORD

1 BYTE = 1 BYTE
1 WORD = 2 BYTE
1 LONGWORD = 4 BYTE
1 QUADWORD = 8 BYTE

QUINDI SPECIFICANDO UNA LETTERA ALLA FINE DI UN'ISTRUZIONE DICO CHE LA SORGENTE E LA DESTINAZIONE LI DEVI INTERPRETARE COME DATI DI 1 BYTE, 2 BYTE, 4 BYTE, 8 BYTE!

IL VALORE DEL REGISTRO DESTINAZIONE È SOVRASCRITTO: SE LO VOGLIO SALVARE DEVO PRIMA FARNE UNA COPIA.

SE GLI OPERANDI DI SORGENTE E DESTINAZIONE SONO DEI REGISTRI, IL NOME DEL REGISTRO CHE ANDREMO AD UTILIZZARE, CAMBIERÀ.

OGNI REGISTRO, AVRÀ UN NOME CHE DIPENDE DALLA TAGLIA IN BYTE DEL DATO CHE

DOBBIAMO ACCEDERE.
IL SUFFISSO DOVRÀ ESSERE COERENTE COL NOME dei REGISTRI SORGENTE e DESTINAZIONE CHE ANDREMO AD UTILIZZARE!

RAX RCX, SONO NOMI DI REGISTRI A 64 BIT.
SE VOGLIO ACCEDERE A RAX A 16 BIT, IL NOME CAMBIERÀ IN "ax".
↓
WORD (W)

E IL NOME DEI REGISTRI VIENE SEMPRE PRECEDUTO DA UN "%".

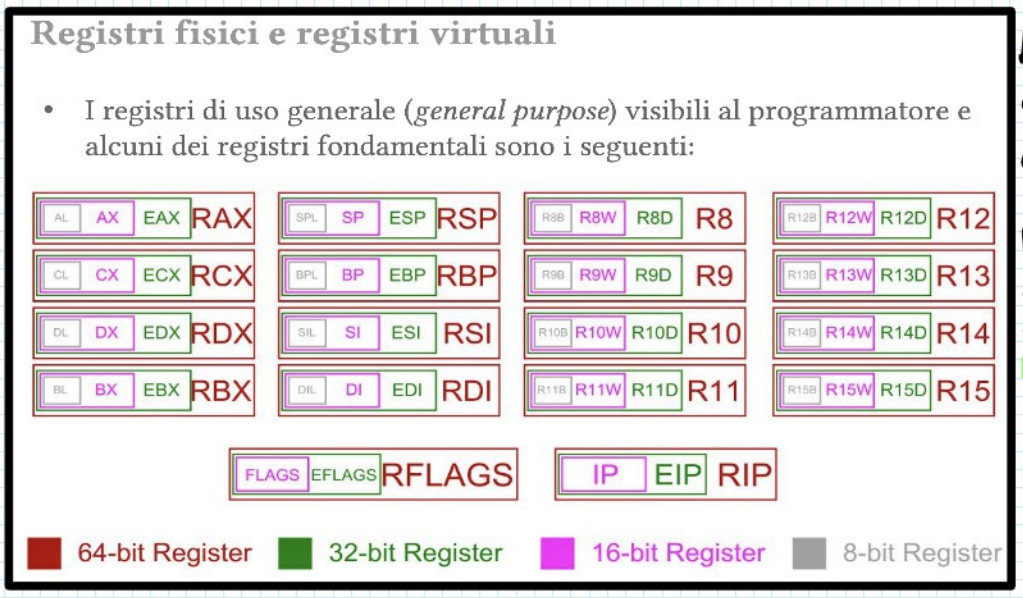
```
ADDW %ax, %bx
```

PRENDI IL REGISTRO "ax" che deve essere UN REGISTRO A 16 bit, PRENDI IL REGISTRO "bx", FANNE LA SOMMA E IL RISULTATO SCRIVILO NEL REGISTRO destinazione.

QUESTA ISTRUZIONE ASSEMBLY VIENE POI PRESA E SCRITTA IN UNA SEQUENZA DI bit che la CPU È IN grado di INTERPRETARE; come? mediante un CIRCUITO COMBINATORIO.

QUALI REGISTRI HO A DISPOSIZIONE?

FACCIAMO UNA DIFFERENZA FRA REGISTRI FISICI E REGISTRI VIRTUALI:



IL REGISTRO fisico È una scatola di flip-flop che ho dentro al mio PROCESSORE;

MA se posso accedere a dati che sono BYTE, LONGWORD, WORD... allora, DI UN REGISTRO fisico leggerò una SOTTOPORZIONE!
↓
REGISTRI VIRTUALI = SONO dei SOTTOINSIEMI che si appoggiano al REGISTRO A 64 bit.

SE IO SCRIVO UN valore a 64 bit dentro un registro fisico E SCRIVO 1 BYTE dentro un REGISTRO VIRTUALE più piccolo E POI LEGGO NUOVAMENTE IL REGISTRO fisico grande, IL MIO VALORE sarà cambiato. PERCHÉ quel BYTE avrà modificato il BYTE meno significativo del mio REGISTRO fisico;

SE SI MODIFICA UN BYTE del REGISTRO FISICO A 64 BIT, IL VALORE PRECEDENTE del REGISTRO A 64 BIT È PERSO.

I REGISTRI FISICI A 64 BIT SONO: RAX, RCX, RDX, RBX, RSP, RBP, RSI, RDI, R8, R9, R10, R11, R12, R13, R14, R15.

COME FACCIO AD ACCEDERE alle SOTTOPORZIONI DEI REGISTRI VIRTUALI?

IN R8 FISICO, PER ACCEDERE alla longword useremo R8D, PER accedere alla WORD R8W, PER ACCEDERE ad 1 BYTE R8B.

CON UN SUFFISSO COERENTE.

$S \in \{B, W, L, Q\}$

BYTE, WORD, LONGWORD, QUADWORD

1 BYTE = 1 BYTE
1 WORD = 2 BYTE
1 LONGWORD = 4 BYTE
1 QUADWORD = 8 BYTE

MA, DEVO POTER COPIARE DATI DA UN REGISTRO ALL'ALTRO E DEVO AVERE UN'INTERCONNESSIONE FRA REGISTRI E POI DEVO INTERCONNETTERE QUESTI REGISTRI AI CIRCUITI CHE FANNO CALCOLI, PER ESEGUIRE LE MIE OPERAZIONI. OSSIA LA ALU e LO SHIFTER.

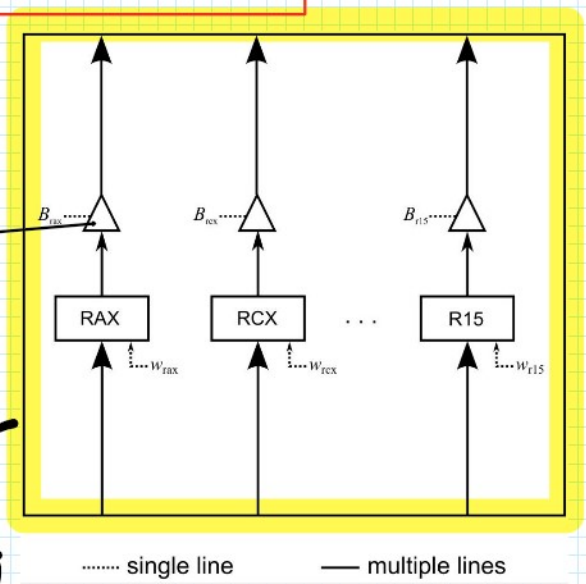
(DATA BUS INTERNO)

INTERCONNESSIONE TRAMITE BUS

- È un BUS interno al processore usato per collegare i registri interni
- Operazioni che caratterizzano il BUS:
  - *ricezione dati*: i bit presenti sul BUS sono memorizzati in un registro
  - *trasmissione dati*: il contenuto di un registro è posto sul BUS
- Al più un solo registro può scrivere sul BUS
- Utilizzo di *segnali di controllo* opportunamente generati:
  - Write enable
  - Buffer Three-State

- $W_i = 1$: si può leggere dal BUS
- $B_i = 1$: si può scrivere sul BUS

- Si può avere una sola scrittura per volta sul BUS
- Pertanto, se si hanno n registri, si dovranno prevedere $2n$ segnali di controllo



Wi è un segnale di controllo che "ANDRÀ" IN AND CON IL clock, se LO ABILITO SU RAX, IL CONTENUTO di R15 COMINCIA a viaggiare SUL BUS, RAGGIUNGE TUTTI QUANTI I REGISTRI, MA POICHÈ SOLO RAX HA Wi=1, aggiornerà IL SUO VALORE.

64 FILI GIRANO INTORNO ALLA CPU.

QUESTA È LA INTERCONNECTION STRUCTURE.

LA ABBIAMO REALIZZATA!