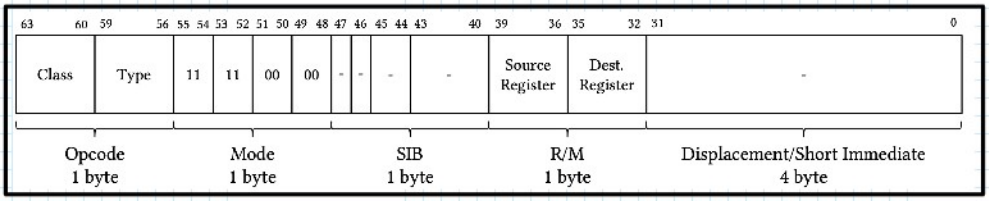
Istr.Logic. Aritmetiche - funzionamento

lunedì 27 febbraio 2023 17:42

pzo >30

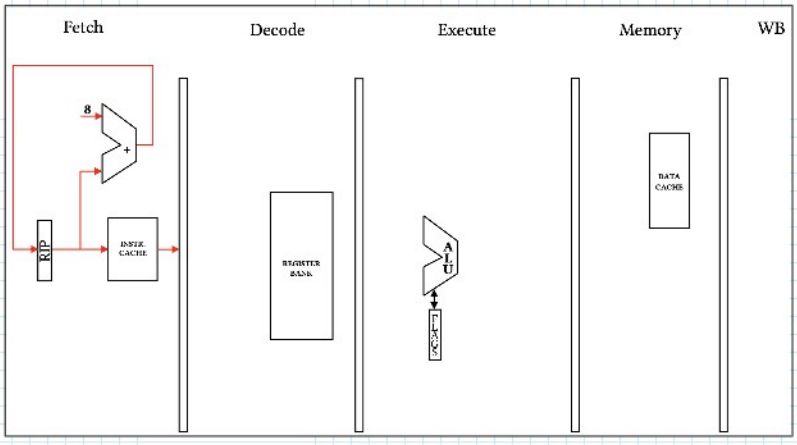
# Istruzioni L/A: formato

1. Non eseguiamo operazioni in memoria, quindi sorgente e destinazione sono registri. Ciò vuol dire che il byte S.I.B è una "don't care conditions".
2. Displacement è anch'essa una "don't care conditions": non utilizziamo né costanti e né effettuiamo operazioni in memoria.
3. Prevediamo di usare solo operandi a 64 bit, quindi Source Size (SS) e Destination Size (DS) saranno entrambi uguali a 1.
4. DI = 0, perchè non abbiamo né spiazzamento né dati immediati;
5. Mem = 00, perchè entrambi gli operandi sono registri;




Quello che ci interessa realmente è l'OPCODE per supportare la fase di DECODE e i registri sorgente e destinazione.

Vediamo ora come organizzare gli stadi della PIPELINE per supportare l'esecuzione di una istruzione logica aritmetica;




Nella fase di Fetch dobbiamo accedere all'INSTRUCTION cache per leggere il contenuto di memoria a 64 bit a partire dall'indirizzo contenuto in RIP e dobbiamo incrementare RIP.
Abbiamo detto che non possiamo utilizzare la ALU perchè è in un altro stadio, devo per forza aggiungere un sommatore. Ho un sommatore che somma 8 all'interno di RIP, e il valore viene scritto all'interno del registro.

DOVE SALVO LA RAPPRESENTAZIONE BINARIA DELLA MIA ISTRUZIONE? Non ho più un IR dedicato, LA SALVO NEL REGISTRO DI PIPELINE.

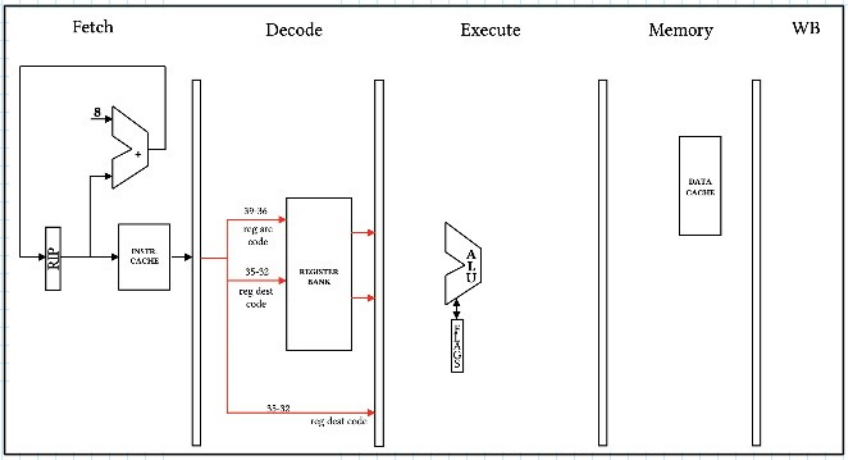
IL REGISTRO DI PIPELINE CONTERRÀ AL SUO INTERNO la RAPPRESENTAZIONE BINARIA DELLA MIA ISTRUZIONE!

decido, la stessa nel registro di pipeline.

IL REGISTRO DI PIPELINE CONTERRÀ AL SUO INTERNO la RAPPRESENTAZIONE BINARIA DELLA MIA ISTRUZIONE!

Cosicché nello stato successivo (decode), la mia unità di CONTROLLO potrà andare a leggere il contenuto di quei bit e capire che cosa deve andare ad eseguire!

Ora passiamo alla decodifica:




ABBIAMO BISOGNO DEL REGISTRO SORGENTE E DEL REGISTRO DESTINAZIONE. TIRIAMO FUORI dal REGISTRO di INTERFACCIA i codici DEL REGISTRO SORGENTE e DEL REGISTRO DESTINAZIONE E LI PASSIAMO AL **BANCO DEI REGISTRI**.

SE PASSO AL BANCO dei REGISTRI questi codici ATTIVANDO la lettura del CONTENUTO, allora io leggo IL CONTENUTO di quei REGISTRI, PERCHÈ io DEVO eseguire un'OPERAZIONE LOGICO-ARITMETICA, quindi UNA SOMMA, se SCRIVO ADD RAX, RBX SOMMO IL CONTENUTO di RAX+RBX E IL RISULTATO lo SCRIVO IN RBX. OSSIA IL REGISTRO DESTINAZIONE.

Ho BISOGNO di TIRARE FUORI dal BANCO dei REGISTRI, IL VALORE CONTENUTO NEI REGISTRI SORGENTE e DESTINAZIONE.

Glieli PASSO E SALVO IL VALORE NEL REGISTRO di **PIPELINE**. Quindi NEL REGISTRO di pipeline ho IL CONTENUTO del REGISTRO SORGENTE e del REGISTRO DESTINAZIONE.

IL RISULTATO PERÒ VA SCRITTO NEL REGISTRO DESTINAZIONE.

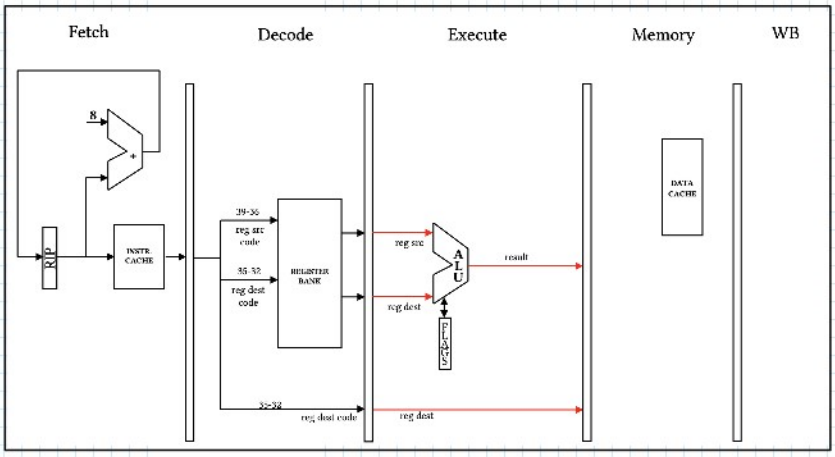
**IN QUALE STADIO ANDRÒ AD EFFETTUARE LA SCRITTURA?** NELLO STADIO di WRITE BACK.

Quindi significa che quando questa ISTRUZIONE ARRIVERÀ nello STADIO di WRITE BACK, quello STADIO DOVRÀ NECESSARIAMENTE SAPERE IL codice del REGISTRO DESTINAZIONE IN cui SCRIVERE IL RISULTATO.

Ma questa INFORMAZIONE, che è SCRITTA NEL REGISTRO di pipeline, VERRÀ SOVRASCRITTA.

↳ quando faccio il fetch dell'ISTRUZIONE successiva.

**QUINDI PROPAGO IN AVANTI QUESTA INFORMAZIONE:**




IL CODICE DEL REGISTRO DESTINAZIONE VIENE COPIATO ALL'INTERNO NEL REGISTRO TAMPONE. POSSO PROPAGARE QUESTO CODICE E FARLO ARRIVARE FINO IN FONDO FINO A QUANDO NON DEVO EFFETTUARE LA SCRITTURA.

IL codice del REGISTRO DESTINAZIONE me lo COPIO NEL REGISTRO TAMPONE!

A QUESTO PUNTO devo eseguire la MIA ISTRUZIONE logico – ARITMETICA.

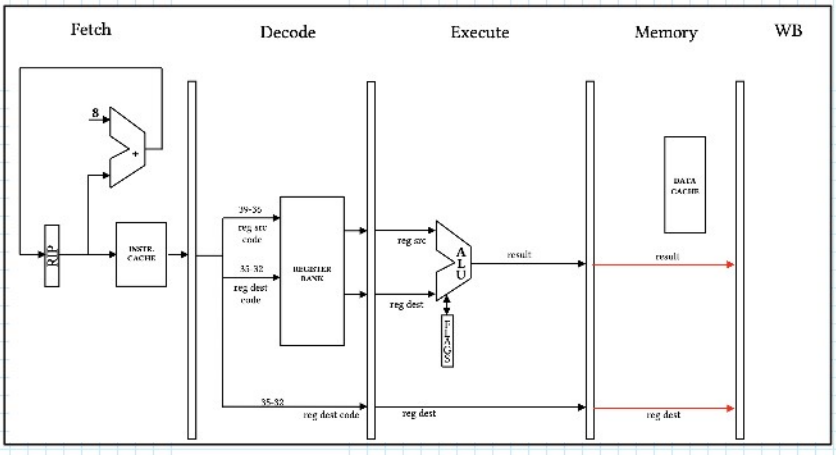
Così ARRIVO al mio STADIO di EXECUTE.

**PASSO ALLA ALU IL REGISTRO SORGENTE E IL REGISTRO DESTINAZIONE.**

L'UNITÀ di CONTROLLO manderà l'OP-code ALLA ALU, LA ALU calcolerà IL RISULTATO e lo SCRIVE NEL REGISTRO di INTERFACCIA di EXCUTE.

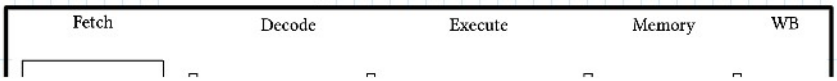
IL CODICE DEL REGISTRO DESTINAZIONE LO PROPAGO IN AVANTI, PERCHÈ NON MI SERVE NELLO STATO DI EXECUTE, MI SERVIRÀ DOPO, IL CONTENUTO VERRÀ SCRITTO QUANDO ARRIVERÒ ALLA FASE DI DECODE DELL'ISTRUZIONE SUCCESSIVA.

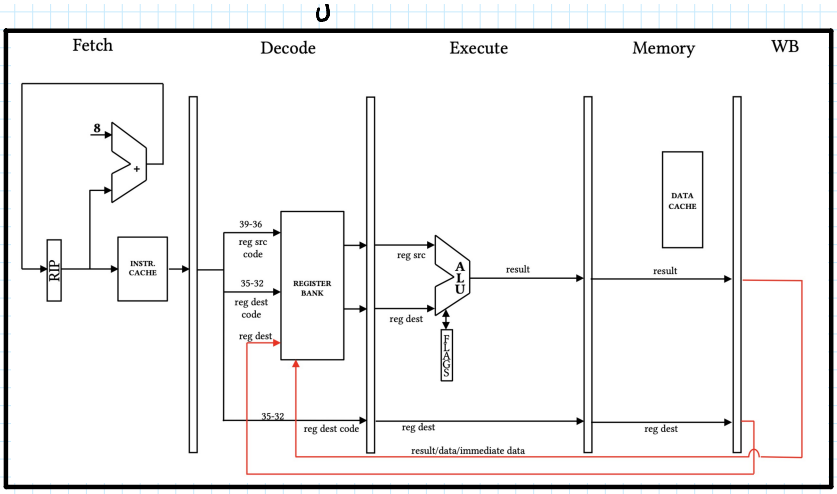
LE ISTRUZIONI LOGICO ARITMETICHE NON ACCEDONO IN MEMORIA, quindi nella fase di MEMORY non facciamo nulla e quindi PROPAGO IL RISULTATO e il codice del REGISTRO DESTINAZIONE.




IN QUESTA FASE SPRECO UN colpo di clock PER NON fare NIENTE! ho soltanto PROPAGATO dei valori che già avevo. NON POSSO SCAVALCARE la fase di memory PERCHÈ nella fase di WB successiva ci potrebbe essere un'altra esecuzione.

**A QUESTO PUNTO** ARRIVA **LA FASE DI WRITEBACK**, in questa fase devo SCRIVERE IL RISULTATO della mia operazione nel BANCO dei REGISTRI.






IL codice del REGISTRO destinazione CE L'HO, IL RISULTATO CE L'HO, ATTIVO IL BANCO DEI REGISTRI!