Introduzione Pipeline

sabato 25 febbraio 2023 17:30

PIPELINE

L'ORGANIZZAZIONE MULTICICLO, SIN ORA ADOTTATA, È TUTTA SEQUENZIALE. QUESTO È L'UNICO PROBLEMA. ABBIAMO AL PIÙ UNA SINGOLA OPERAZIONE IN ESECUZIONE SUL PROCESSORE, PER VOLTA.

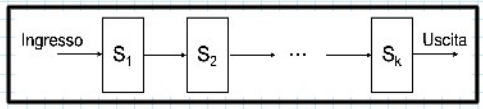
Motivazioni per una nuova architettura

- L'organizzazione a multiciclo è efficace perché permette il riuso dell'hardware per svolgere compiti differenti
- Tuttavia è un'organizzazione completamente *sequenziale*
  - In un istante di tempo, c'è una sola istruzione (meglio: microoperazione) in esecuzione nella CPU (*in-flight operation*)

• IL PIPELINING

A DIFFERENZA DELLE MICROOPERAZIONI, IN CUI UNA SINGOLA ISTRUZIONE VENIVA SPEZZETTATA IN PIÙ PASSI ELEMENTARI, ORGANIZZO L'HARDWARE DEL MIO PROCESSORE IN PASSI DIFFERENTI D'ESECUZIONE, OSSIA IN STADI DELLA PIPELINE.

- È una tecnica per progettare un'architettura hardware che:
  - consenta il miglioramento delle prestazioni del processore
  - si basa sulla *sovrapposizione* dell'esecuzione di *più istruzioni* appartenenti ad un flusso di esecuzione sequenziale
- Il lavoro svolto dal processore è suddiviso in *passi* (*stadi della pipeline*) che richiedono una frazione del tempo d'esecuzione di un'intera istruzione
- Gli stadi sono connessi sequenzialmente per formare la pipeline
- Le istruzioni vengono elaborate nei vari stadi secondo l'ordine previsto



CERCHIAMO DI MIGLIORARE LE PRESTAZIONI DEL PROCESSORE CERCANDO DI AVERE PIÙ IN-FLIGHT INSTRUCTION ALLO STESSO TEMPO.

ALLO STESSO ISTANTE DI TEMPO, IL MIO PROCESSORE PERMETTE DI AVERE PIÙ ISTRUZIONI IN ESECUZIONE IN STADI DIFFERENTI.

HO UN'ISTRUZIONE CHE ENTRA E QUESTA VIENE PARZIALMENTE ESEGUITA NELLO STADIO 1, POI QUESTA ISTRUZIONE PARZIALMENTE ESEGUITA VIENE PASSATA NELLO STADIO 2, MA INTANTO LO STADIO 1 È LIBERO QUINDI POSSO TIRARMI DENTRO L'ISTRUZIONE SUCCESSIVA. E COSÌ VIA...

MI TIRO DENTRO ALLO STADIO PRECEDENTE L'OPERAZIONE SUCCESSIVA:

ESEMPIO BUCATO

- Attività elementari per fare il bucato:
  - Lavaggio
  - Asciugatura
  - Stiratura
  - Riordino

ESECUZIONE SEQUENZIALE →

ESECUZIONE PIPELINE →

PER PROBLEMI DI TEMPO NON VEDREMO TUTTO LO Z64 IN PIPELINE, MA QUELLO CHE SI FARÀ SARÀ PRENDERE UN SOTTOINSIEME DI ISTRUZIONI PER VEDERE COME POSSIAMO ORGANIZZARE IL NOSTRO PROCESSORE PER CONSENTIRE L'ESECUZIONE DI QUESTE ISTRUZIONI IN MODALITÀ PIPELINE.

LE ISTRUZIONI CHE ANDREMO A CONSIDERARE SONO:

- NOP
- IST. LOGICO ARITMETICHE
- IST. LOAD/STORE
- JUMP CONDIZIONATO

Class	Instruction	Syntax	Semantics
HW	Non-operational	`nop`	No operation (beyond the fetch phase)
L/A	Sum	`addq %regsorg, %regdest`	regdest = regsorg + regdest
	Subtraction	`subq %regsorg, %regdest`	regdest = regdest - regsorg
	Logic product	`andq %regsorg, %regdest`	regdest = regsorg and regdest
	Logic sum	`orq %regsorg, %regdest`	regdest = regsorg or regdest
	Logic negation	`notq %register`	register = not register
L/S	Loading of word	`movq offset(%regbase), %regdest`	regdest = memory[offset+regbase]
	Storage of word	`movq %regsorg, offset(%regbase)`	memory[offset+regbase] = regsorg
J	jump if flag X == 1	`jX displacement`	if flag X == 1 then RIP = RIP + displacement