Evoluția arhitecturilor calculatoarelor (2)

Author: Petre IORDANESCU, Date: January 2021
Categories: Software Design & Architectures

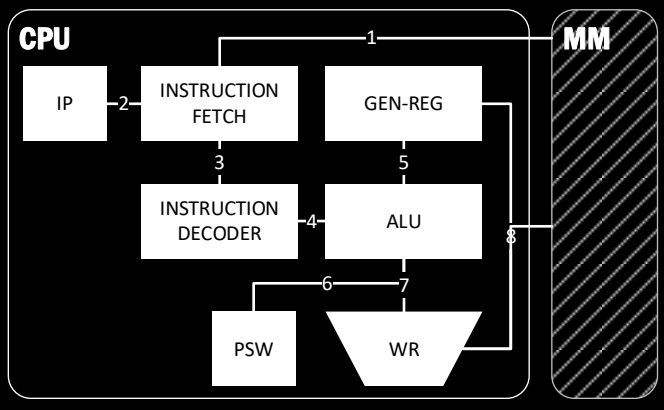
ARHITECTURA GENRALA A UNUI CALCULATOR

Un calculator general este format din urmatoarele elemente:

- Unitatea centrala de procesare (procesorul) numita **CPU** (Central Processing Unit), acronim încetățenit si care va folosit ca atare. Acest element este direct răspunzător de execuția operațiilor.
- Memoria centrala numita **MM** (Main Memory). Este non-persistenta[1] dar are viteza de acces[2] mare.
- Memoria secundara reprezentata de discuri si face parte din „setul" de dispozitive periferice. Este caracterizata de viteza de acces medie-mare (depinde tipul de discuri) si de faptul ca este persistenta.
- Doua dispozitive periferice esentiale, unul de introducere (tastatura) si unul de afișare (monitor) numite impreuna **CONSOLA**.

Articolul prezent (si in general aceasta serie de articole) se refera la arhitectura CPU. Prin „calculator" se face de fapt referire la CPU din rațiuni de ordin tehnic, mai exact pentru a putea fi inteles si de non-specialiști in domeniu. Sa vedem așadar arhitectura generala a **CPU**, fara a avea absolut nici o pretenție de detalii tehnice si de alte componente interioare – scopul este de a intelege, de ce si cum funcționează:

- IP reprezintă adresa de memorie de unde se va încărca o instrucțiune spre a fi executata; dupa încărcarea instrucțiunii, uzual acesta este incrementat si pregatit pentru urmatoarea instructiune;
- INSTRUCTION FETCH aduce o instructiune din memorie de la adresa din IP, o da către INSTRUCTION DECODER si incrementează IP;
- INSTRUCTION DECODER va decodifică instrucțiunea primita si va separa câmpurile acesteia: codul operatiei (OP-CODE) si adresele registrelor conținute in instrucțiune;
- ALU preia OP-CODE si adresele registrelor, aduce datele din registre din modulul GEN-REG (registrele generale), executa operatia ceruta prin OP-CODE si transmite rezultatul către WR;
- PSW (Program Status Word) este un registru special unde sunt conținute informații despre starea masinii si ultimul rezultat **ALU** (de exemplu daca rezultatul a fost zero, pozitiv sau negativ daca a generat o depășire[3], etc;
- WR va scrie acest rezultat inapoi in GEN-REG sau MM in functie de cerinta instrucțiunii;



ARHITECTURA MONOLIT

Asa cum au fost prezentate lucrurile, totul „pare" un monolit: exista o secventa stricta si precisa ca ordine in are se executa pașii pentru a finaliza o instrucțiune. Acesta se numeste *„ciclu instrucțiune*[4]". Durata lui depinde de tipul de instrucțiune executata si se măsoară in ‚cicluri de ceas masina". Astfel, in schemele si implementările monolit, sau cu modulele prezentate funcționând in secventa (si nu independente), un ciclu instrucțiune este egal cu un ciclu de ceas. Si datorita faptului ca „traversarea" tuturo componentelor electronice de mai sus (asta sunt, nu? Sper ca ati realizat acest lucru...) necesita un timp strict mai mare ca zero (semnalul electric circula cu o viteza finita..., nu insist). Din acest motiv, ceasul fizic este imitat superior ca frecventa, limita fiind determinata de proiectanti iar depasirea ei (over-clocking) poate duce la rezultate impredictibile (iar nu insist; daca nu se intelege de ce atunci trebuiesc si alte articole despre aceasta tema).

Si uite asa am ajuns intr-un punct in care limitarea fizica nu poate fi depășită si trebuiesc gasite alte metode de crestere a performantei.

Cea uzuala (in foarte multe alte domenii) este creșterea productivității, mai neaoș spus „daca un om sapa un șanț într-o zi, atunci doi oameni il termina in jumatate de zi". Pentru a putea realiza acest lucru trebuie ca modulele din CPU sa poata lucra independent intre ele, la viteze diferite (relativ asincron). Realizarea acestor deziderate este posibila prin introducerea de registre intre componente (cu rol de memorare a *ultimului* rezultat al fiecărei componente individuale. Astfel se va putea realiza o arhitectura de tip PIPELINE (sau banda de asamblare), arhitecturi ce vor fi studiate in următorul numar.

RREFETINTE BIBLIOGRAFICE

- *CS 61C. Great Ideas in Computer Architecture. (Machine Structures). Lecture: Single-Cycle CPU. Datapath Design. Guest Lecturer TA: Shreyas Chand*. https://inst.eecs.berkeley.edu/~cs61c/fa14/lec/27/2014Fa-CS61C-L27-sc-CPU-1up.pdf
- *Single Cycle CPU:* https://cseweb.ucsd.edu/~j2lau/cs141/week3.html
- *Single Cycle CPU:* https://cseweb.ucsd.edu/classes/wi13/cse141-b/slides/05-SingleCycleCPU.pdf
- A *single-cycle MIPS processor*: https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec08.pdf
- *MIPS Single-Cycle Processor Implementation:* https://www.d.umn.edu/~gshute/mips/single-cycle.html

RECOMANDARI

Personal recomand, mai ales celor ce doresc sa afle mai multe despre istoria si evoluția calculatoarelor si a arhitecturilor, citirea tuturor articolelor si materialelor referențiate in notele de subsol, ele fiind un material bibliografic inestimabil si care pune in evidenta modul si evoluția gindirii oamenilor.

RENware team si Petre IORDANESCU, 2020

(mai multe articole gasiti pe: http://www.renware.eu/articles)

[1] Persistenta defineste capabilitatea unui dispozitiv de a pastra datele dupa oprirea tensiunii
[2] Prin acces se intelege oricare din operatiile de citire sau scriere
[3] Cary sau prescurtat CY
[4] Instruction Cycle