NOTIȚE CURS 0x07.08

Source code (C, Java, ...) —compile→ ASSEMBLY —asamblăm→ Object file —link→ Binary file → procesare

Object file + libraries (pt. eficientizare, funcții, ...) —link→ Binary file

ISA - instruction set architecture : reg., instrucțiuni, tipuri de date, metode de adresare a memoriei

PIPELINING = un tip de paralelism la nivel de instrucțiune

↳ nu putem face asta mereu deoarece apar pipeline stalls (întârzieri în conducta de date) și hazards (riscuri)

1) structural hazards = o unit de calcul este deja utilizată iar 2 instr. încearcă să acceseze ac. unitate

2) data hazards : datele nu sunt reg. pt. utilizare o instr. depinde de rezultatul unei instr. precedente

3) control hazards: nu știm următoarea instr., din cauza unei instr. de jmp. nu știm următoarea instr.

True Dependence (Read after write: RAW)

I add %ebx, %eax (write) ecx = (ebx + eax) ← să facă înainte să i se seteze noua val.

II sub %eax, %ecx (read)

Anti-Dependence (Write After Read: WAR)

s-a putea: ecx - eax, eax + ebx

add %ebx, %eax → depinde

sub %ecx, %ebx → scrie

eax = eax + (ebx) vechi sau nou? → să-l las să scrie înainte să se fol. de vechia val.

ebx = ebx - ecx

Output Dependence (Write After Write: WAW)

mov $0x10, %eax } val care va fi în eax

mov $0x01, %eax

! sid. se mai complică și din cauza faptului că instr. au nevoie de mai multe cicli de ceas decât de cicli de CPU