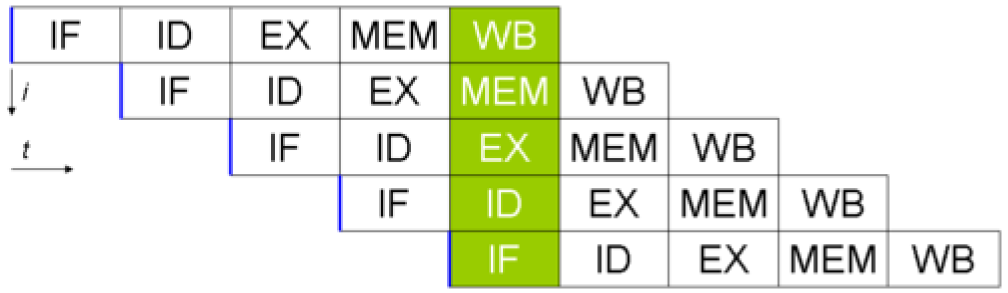


24. RISC Pipelining

Chapter 15
sections 15.5

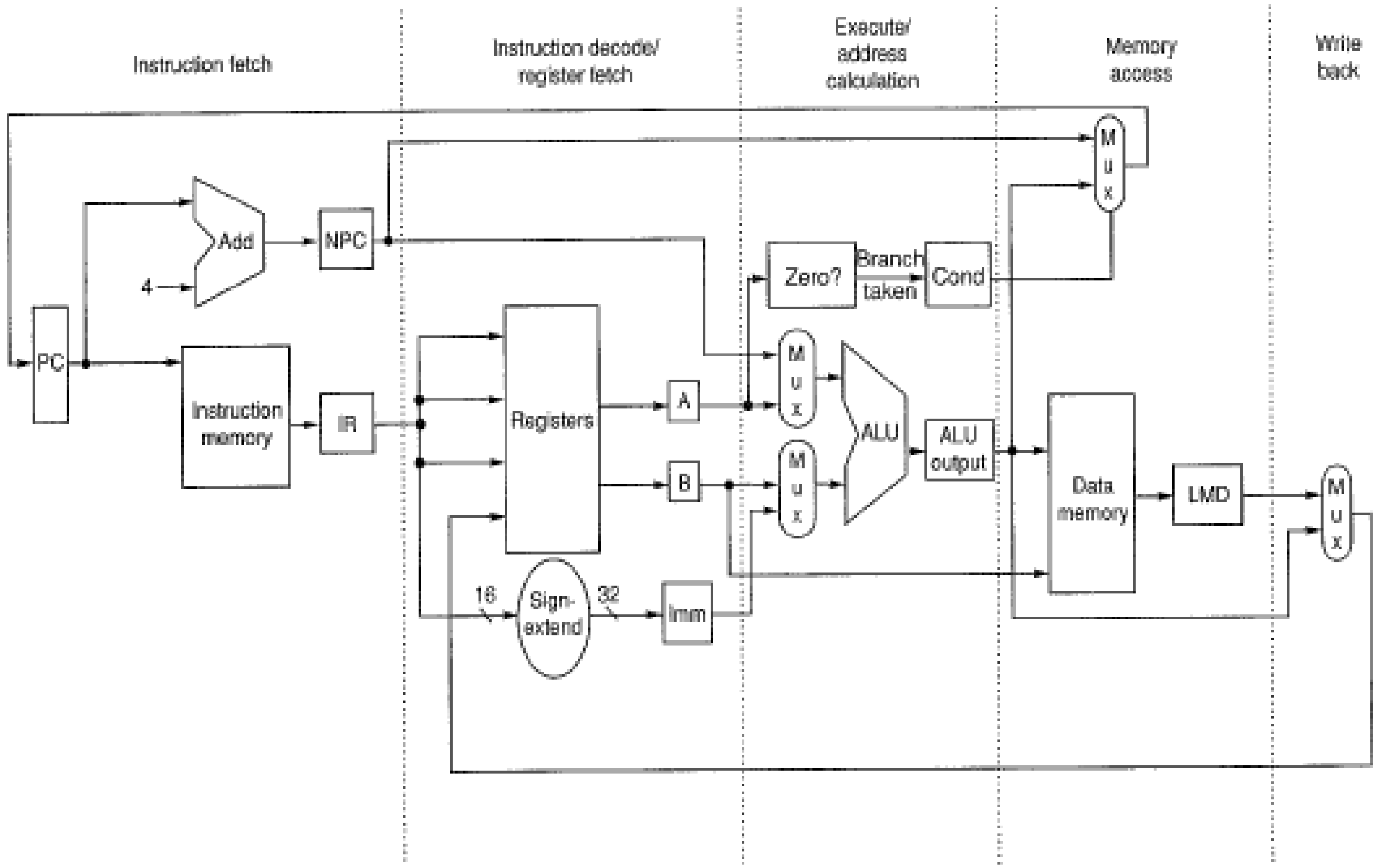
MIPS Instruction Set

- The MIPS instruction set was designed for pipeline execution
    - MIPS instructions are the same length. x86 instructions vary from 1 byte to 17 bytes and pipelining is much more challenging.
    - MIPS has only a few instruction formats, with the source operand being located in the same place in each instruction.
    - Memory operands only appear in loads or stores in MIPS.
    - Operands must be aligned in memory.

MIPS Instructions

- MIPS instructions classically take five steps:
  - Instruction Fetch (IF)
  - Instruction Decode & Register Fetch (ID)
  - Execution / Effective Address (EX)
  - Memory Access (MEM)
  - Write Back Results (WO)

Non-pipelined RISC Processor (MIPS) 5 stages



Instruction Fetch (IF)

- **IF (Instruction Fetch)**
  IR <- M[PC]
  NPC <- PC + 4

- IR: Instruction Register
- M: Memory
- PC: Program Counter
- NPC: Next Program Counter

- Fetch the instruction
- Update program counter

Instruction Decode (ID)

- **ID (Instruction Decode/Register Fetch)**
  - A <- Regs[rs]
  - B <- Regs[rt]
  - Imm<- sign-extend immediate field of IR

- Decoding can be done in parallel with reading registers

- In an aggressive implementation the branch can be completed at the end of this stage as we will see later

Execution (EX)

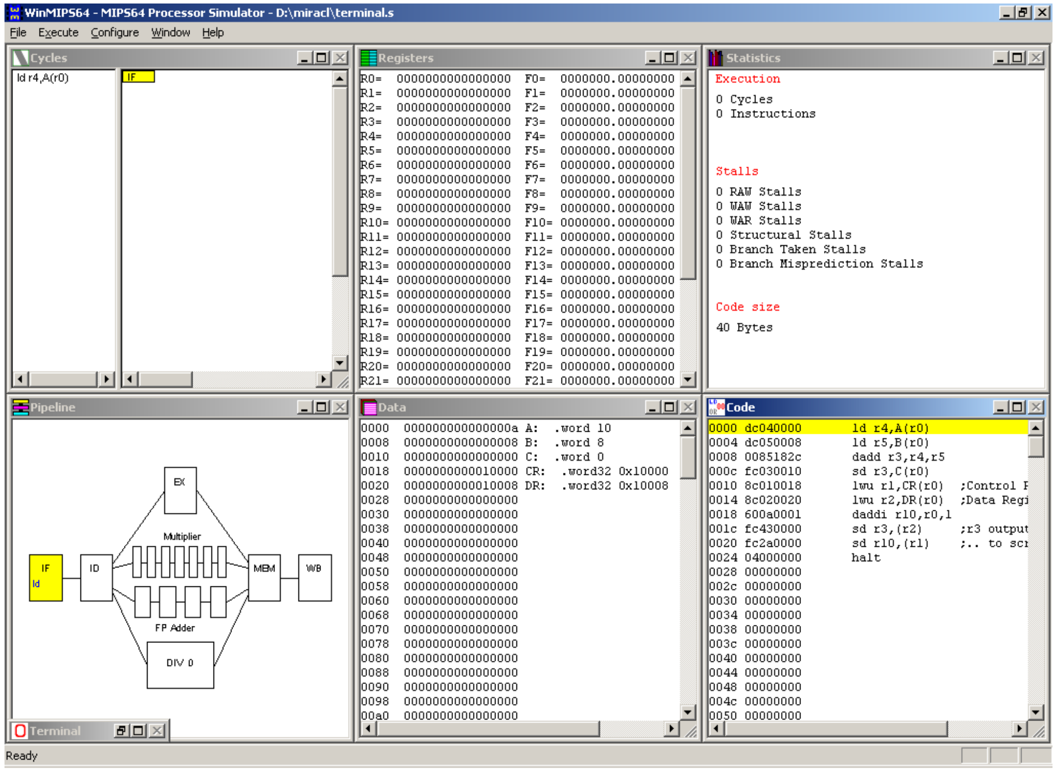
- Performs one of the following:
  - Memory Reference
    - ALUOutput <- A + immediate
  - Register-Register ALU instruction
    - ALUOutput<- A opcode B
  - Register-Immediate ALU instruction
    - ALUOutput<- A opcode Imm
  - Branch
    - ALUOutput<- NPC + (Imm << 2)
    - Cond <- (A==0)

Memory Access (MEM)

- Memory Reference
  - LMD <- Mem[ALUOutput] ; load from memory
  - Mem[ALUOutput] <- B ; store to memory
- Branch
  - if (cond) PC <- ALUOutput

Write Back (WB)

- Register-Register ALU
  - Regs[rd] <- ALUOutput
- Register-Immediate ALU
  - Regs[rt] <- ALUOutput
- Load instruction
  - Regs[rt] <- LMD


WinMIPS64 - MIPS64 Processor Simulator - D:\miracl\terminal.s
File Execute Configure Window Help
Cycles
Registers
Statistics
Pipeline
Data
Code
Terminal
IF
Id
ID
EX
Multiplier
FP Adder
DIV 0
MEM
WB
Ready

```
ld r4,A(r0)    IF
```

```
R0=  0000000000000000  F0=  0000000.00000000
R1=  0000000000000000  F1=  0000000.00000000
R2=  0000000000000000  F2=  0000000.00000000
R3=  0000000000000000  F3=  0000000.00000000
R4=  0000000000000000  F4=  0000000.00000000
R5=  0000000000000000  F5=  0000000.00000000
R6=  0000000000000000  F6=  0000000.00000000
R7=  0000000000000000  F7=  0000000.00000000
R8=  0000000000000000  F8=  0000000.00000000
R9=  0000000000000000  F9=  0000000.00000000
R10= 0000000000000000  F10= 0000000.00000000
R11= 0000000000000000  F11= 0000000.00000000
R12= 0000000000000000  F12= 0000000.00000000
R13= 0000000000000000  F13= 0000000.00000000
R14= 0000000000000000  F14= 0000000.00000000
R15= 0000000000000000  F15= 0000000.00000000
R16= 0000000000000000  F16= 0000000.00000000
R17= 0000000000000000  F17= 0000000.00000000
R18= 0000000000000000  F18= 0000000.00000000
R19= 0000000000000000  F19= 0000000.00000000
R20= 0000000000000000  F20= 0000000.00000000
R21= 0000000000000000  F21= 0000000.00000000
```

```
Execution
0 Cycles
0 Instructions

Stalls
0 RAW Stalls
0 WAW Stalls
0 WAR Stalls
0 Structural Stalls
0 Branch Taken Stalls
0 Branch Misprediction Stalls

Code size
40 Bytes
```

```
0000  000000000000000a A:   .word 10
0008  0000000000000008 B:   .word 8
0010  0000000000000000 C:   .word 0
0018  0000000000010000 CR:   .word32 0x10000
0020  0000000000010008 DR:   .word32 0x10008
0028  0000000000000000
0030  0000000000000000
0038  0000000000000000
0040  0000000000000000
0048  0000000000000000
0050  0000000000000000
0058  0000000000000000
0060  0000000000000000
0068  0000000000000000
0070  0000000000000000
0078  0000000000000000
0080  0000000000000000
0088  0000000000000000
0090  0000000000000000
0098  0000000000000000
00a0  0000000000000000
```

```
0000 dc040000        ld r4,A(r0)
0004 dc050008        ld r5,B(r0)
0008 0085182c        dadd r3,r4,r5
000c fc030010        sd r3,C(r0)
0010 8c010018        lwu r1,CR(r0)   ;Control F
0014 8c020020        lwu r2,DR(r0)   ;Data Regi
0018 600a0001        daddi r10,r0,1
001c fc430000        sd r3,(r2)      ;r3 output
0020 fc2a0000        sd r10,(r1)     ;.. to scr
0024 04000000        halt
0028 00000000
002c 00000000
0030 00000000
0034 00000000
0038 00000000
003c 00000000
0040 00000000
0044 00000000
0048 00000000
004c 00000000
0050 00000000
```

WINMIPS64

WinMIPS64

- WinMIPS64 is an instruction set simulator

- You can:
  - Load MIPS programs
  - Execute one cycle at a time
  - Visualize the pipeline

- You can download the software and read the documentation here:
  - http://indigo.ie/~mscott/

WinMIPS64 Example

- Open notepad, copy the following into a new notepad file, and save it as test.s
- Copy WinMIPS64 from the CS 430 public folder
- Open WinMIPS64
- From WinMIPS64, open test.s
- Step through the program using F7