Reduced Instruction Set Computer

Virendra Singh

Associate Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

Computer Organization & Architecture

Lecture 10 (08 April 2013)

CADSL

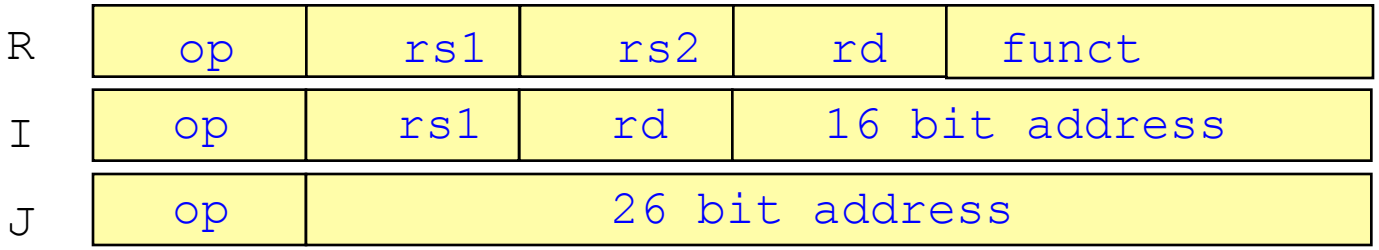
Instruction Set Summary

- Instruction complexity is only one variable
  - lower instruction count vs. higher CPI / lower clock rate

- Design Principles:
  - simplicity favors regularity
  - smaller is faster
  - good design demands compromise
  - make the common case fast
- Instruction set architecture
  - a very important abstraction indeed!

Overview of DLX

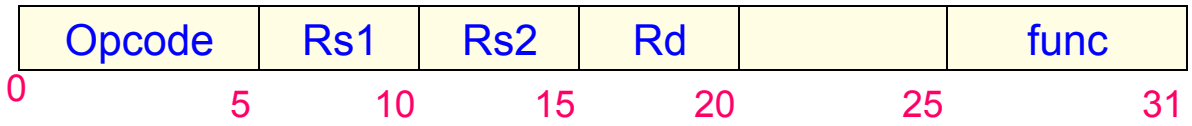
- simple instructions, all 32 bits wide
- very structured, no unnecessary baggage
- only three instruction formats



- rely on compiler to achieve performance

Instruction Set

Register-Register Instructions



Arithmetic and Logical Instruction

- ADD Rd, Rs1, Rs2 Regs[Rd] <= Reg[Rs1] + Reg[Rs2]
- SUB Rd, Rs1, Rs2 Regs[Rd] <= Reg[Rs1] - Reg[Rs2]
- AND Rd, Rs1, Rs2 Regs[Rd] <= Reg[Rs1] and Reg[Rs2]
- OR Rd, Rs1, Rs2 Regs[Rd] <= Reg[Rs1] or Reg[Rs2]
- XOR Rd, Rs1, Rs2 Regs[Rd] <= Reg[Rs1] xor Reg[Rs2]
- SUB Rd, Rs1, Rs2 Regs[Rd] <= Reg[Rs1]-Reg[Rs2]

DLX Instruction Set

ADD Rd, Rs1, Rs2	Rd ← Rs1 + Rs2 (overflow – exception)	R	000_000 000_100
SUB Rd, Rs1, Rs2	Rd ← Rs1 - Rs2 (overflow – exception)	R	000_000 000_110
AND Rd, Rs1, Rs2	Rd ← Rs1 and Rs2	R	000_000/ 001_000
OR Rd, Rs1, Rs2	Rd ← Rs1 or Rs2	R	000_000/ 001_001
XOR Rd, Rs1, Rs2	Rd ← Rs1 xor Rs2	R	000_000/ 001_010
SLL Rd, Rs1, Rs2	Rd ← Rs1 << Rs2 (logical) (5 lsb of Rs2 are significant)	R	000_000 001_100
SRL Rd, Rs1, Rs2	Rd ← Rs1 >> Rs2 (logical) (5 lsb of Rs2 are significant)	R	000_000 001_110
SRA Rd, Rs1, Rs2	Rd ← Rs1 >> Rs2 (arithmetic) (5 lsb of Rs2 are significant)	R	000_000 001_111

DLX Instruction Set

ADDI Rd, Rs1, Imm	Rd ← Rs1 + Imm (sign extended) (overflow – exception)	I	010_100
SUBI Rd, Rs1, Imm	Rd ← Rs1 – Imm (sign extended) (overflow – exception)	I	010_110
ANDI Rd, Rs1, Imm	Rd ← Rs1 and Imm (zero extended)	I	011_000
ORI Rd, Rs1, Imm	Rd ← Rs1 or Imm(zero extended)	I	011_001
XORI Rd, Rs1, Imm	Rd ← Rs1 xor Imm(zero extended)	I	011_010
SLLI Rd, Rs1, Imm	Rd ← Rs1 << Imm (logical) (5 lsb of Imm are significant)	I	011_100
SRLI Rd, Rs1, Imm	Rd ← Rs1 >> Imm (logical) (5 lsb of Imm are significant)	I	011_110
SRAI Rd, Rs1, Imm	Rd ← Rs1 >> Imm (arithmetic) (5 lsb of Imm are significant)	I	011_111

DLX Instruction Set

LHI Rd, Imm	Rd(0:15) ← Imm Rd(16:32) ← hex0000 (Imm: 16 bit immediate)	I	011_011
NOP	Do nothing	R	000_000 000_000

DLX Instruction Set

SEQ Rd, Rs1, Rs2	Rs1 = Rs2: Rd ← hex0000_0001 else: Rd ← hex0000_0000	R	000_000 010_000
SNE Rd, Rs1, Rs2	Rs1 /= Rs2: Rd ← hex0000_0001 else: Rd ← hex0000_0000	R	000_000 010_010
SLT Rd, Rs1, Rs2	Rs1 < Rs2: Rd ← hex0000_0001 else: Rd ← hex0000_0000	R	000_000 010_100
SLE Rd, Rs1, Rs2	Rs1 <= Rs2: Rd ← hex0000_0001 else: Rd ← hex0000_0000	R	000_000 010_110
SGT Rd, Rs1, Rs2	Rs1 > Rs2: Rd ← hex0000_0001 else: Rd ← hex0000_0000	R	000_000 011_000
SGE Rd, Rs1, Rs2	Rs1 >= Rs2: Rd ← hex0000_0001 else: Rd ← hex0000_0000	R	000_000 011_010

DLX Instruction Set

SEQI Rd, Rs1, Imm	Rs1 = Imm : Rd ← hex0000_0001 else: Rd ← hex0000_0000 (Imm: Sign extended 16 bit immediate)	I	100_000
SNEI Rd, Rs1, Imm	Rs1 /= Imm : Rd ← hex0000_0001 else: Rd ← hex0000_0000	I	100_010
SLTI Rd, Rs1, Imm	Rs1 < Imm : Rd ← hex0000_0001 else: Rd ← hex0000_0000	I	100_100
SLEI Rd, Rs1, Imm	Rs1 <= Imm : Rd ← hex0000_0001 else: Rd ← hex0000_0000	I	100_110
SGTI Rd, Rs1, Imm	Rs1 > Imm : Rd ← hex0000_0001 else: Rd ← hex0000_0000	I	101_000
SGEI Rd, Rs1, Imm	Rs1 >= Imm : Rd ← hex0000_0001 else: Rd ← hex0000_0000	I	101_010

DLX Instruction Set

BEQZ Rs, Label	Rs = 0: PC ← PC+4+Label Rs /= 0: PC ← PC+4 (Label: Sign extended16 bit immediate)	I	010_000
BNEZ Rs, Label	Rs /= 0: PC ← PC+4+Label Rs = 0: PC ← PC+4	I	010_001
J Label	PC ← PC + 4 + sign_extd(imm26)	J	001_100
JAL Label	R31 ← PC + 4 PC ← PC+ 4 + sign_extd(imm26)	J	001_100
JAL Label	R31 ← PC + 4 PC ← PC+ 4 + sign_extd(imm26)	J	001_101
JR Rs	PC ← Rs	I	001_110
JALR Rs	R31 ← PC + 4 PC ← Rs	I	001_111

DLX Instruction Set

LW Rd, Rs2 (Rs1)	Rd ← M(Rs1 + Rs2) (word aligned address)	R	000_000 100_000
SW Rs2(Rs1), Rd	M(Rs1 + Rs2) ← Rd	R	000_000 101_000
LH Rd, Rs2 (Rs1)	Rd (16:31)← M(Rs1 + Rs2) (Rd sign extended to 32 bit)	R	000_000 100_001
SH Rs2(Rs1), Rd	M(Rs1 + Rs2) ← Rd(16:31)	R	000_000 101_001
LB Rd, Rs2 (Rs1)	Rd (24:31)← M(Rs1 + Rs2) (Rd sign extended to 32 bit)	R	000_000 101_010
SB Rs2(Rs1), Rd	M(Rs1 + Rs2) ← Rd(24:31)	R	000_000 101_010

DLX Instruction Set

LWI Rd, Imm (Rs)	Rd ← M(Rs + Imm) (Imm: sign extended 16 bit) (word aligned address)	I	000_100
SWI Imm(Rs), Rd	M(Rs + Imm) ← Rd	I	001_000
LHI Rd, Imm (Rs)	Rd (16:31)← M(Rs + Imm) (Rd sign extended to 32 bit)	I	000_101
SHI Imm(Rs), Rd	M(Rs1 + Rs2) ← Rd(16:31)	I	001_001
LBI Rd, Imm (Rs)	Rd (24:31)← M(Rs + Imm) (Rd sign extended to 32 bit)	I	000_110
SBI Imm(Rs), Rd	M(Rs + Imm) ← Rd(24:31)	I	001_010

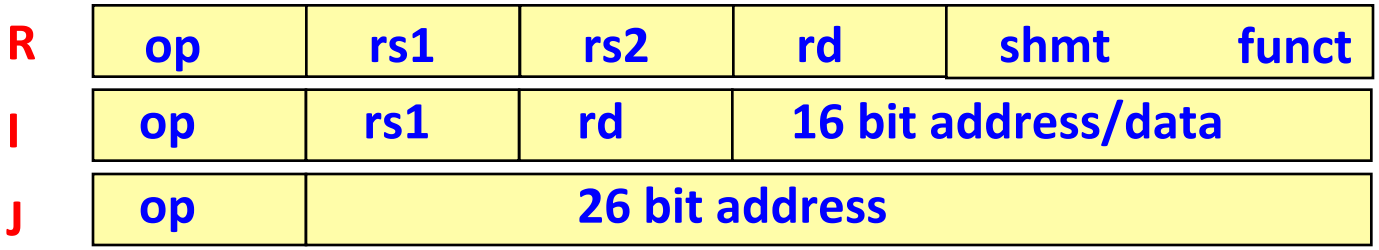
Example Processor MIPS subset

MIPS Instruction – Subset

- Arithmetic and Logical Instructions
  - add, sub, or, and, slt
- Memory reference Instructions
  - lw, sw
- Branch
  - beq, j

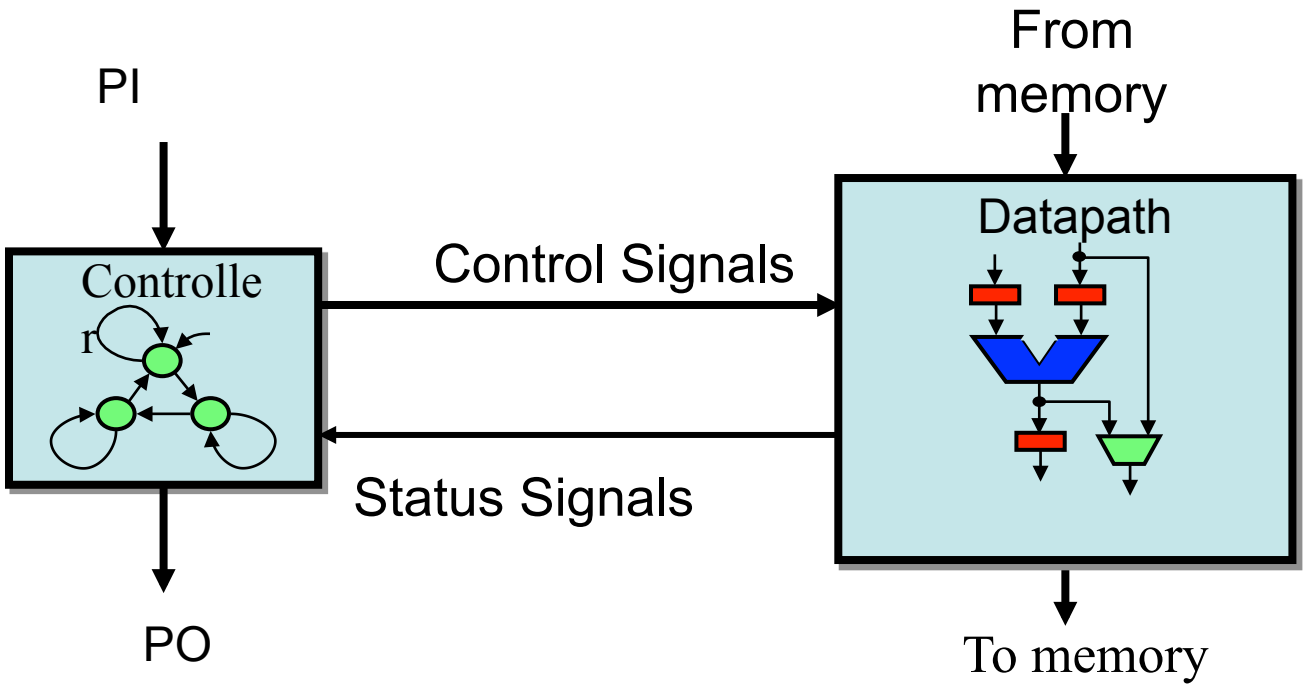
Overview of MIPS

- simple instructions, all 32 bits wide
- very structured, no unnecessary baggage
- only three instruction formats



- rely on compiler to achieve performance

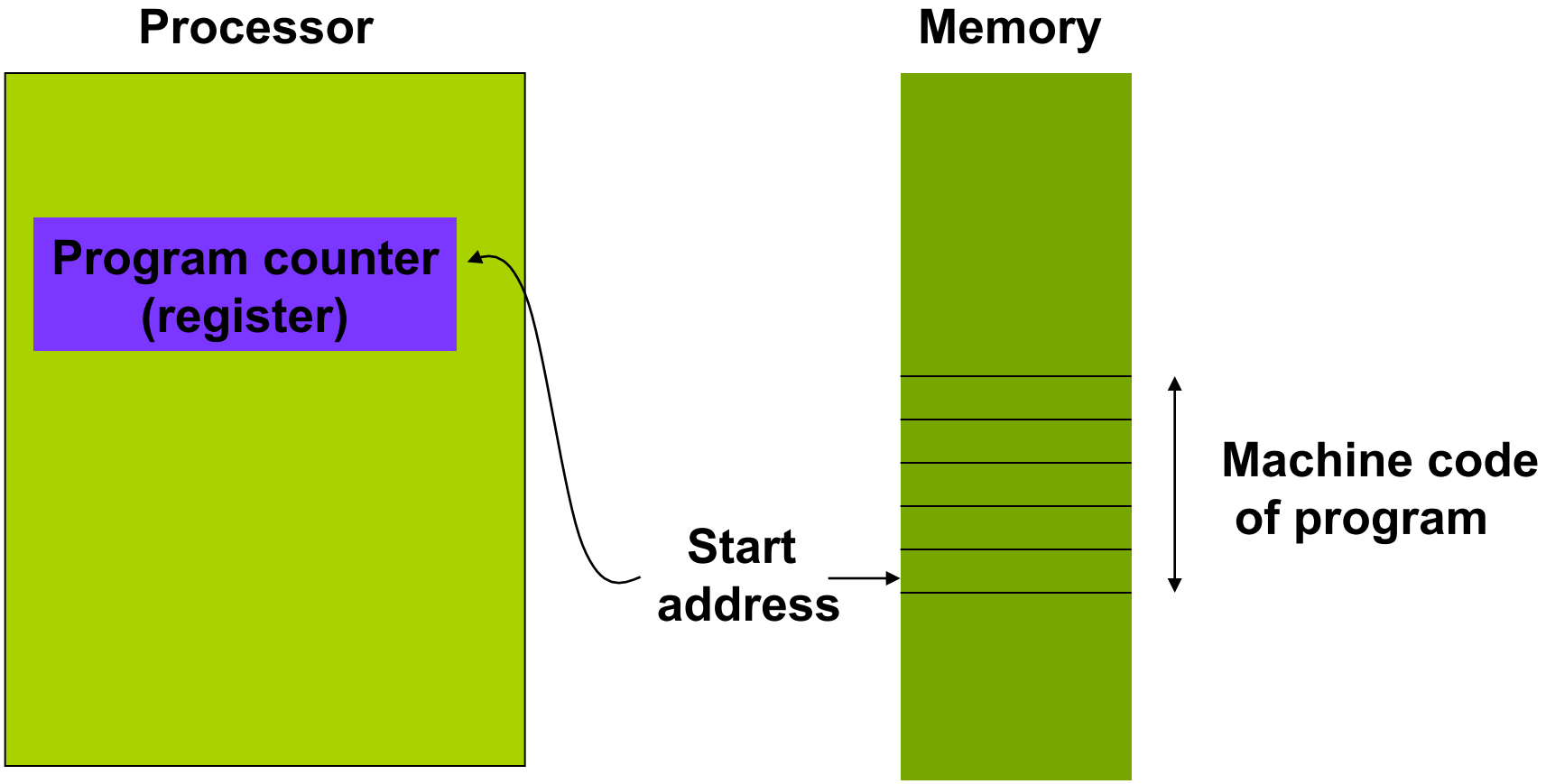
Processor Architecture



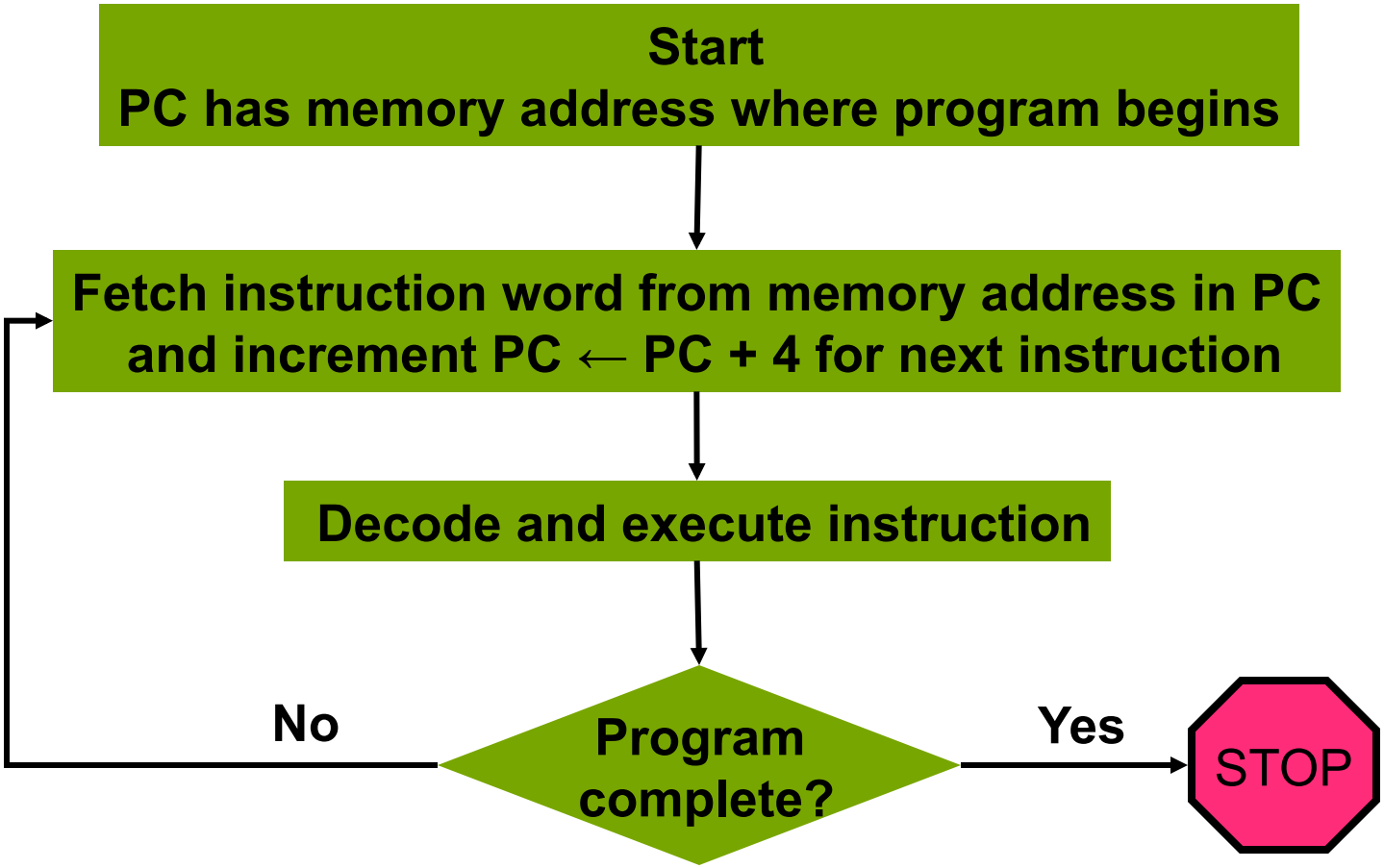
Where Does It All Begin?

- In a register called *program counter (PC).*
- PC contains the memory address of the next instruction to be executed.
- In the beginning, PC contains the address of the memory location where the program begins.

Where is the Program?



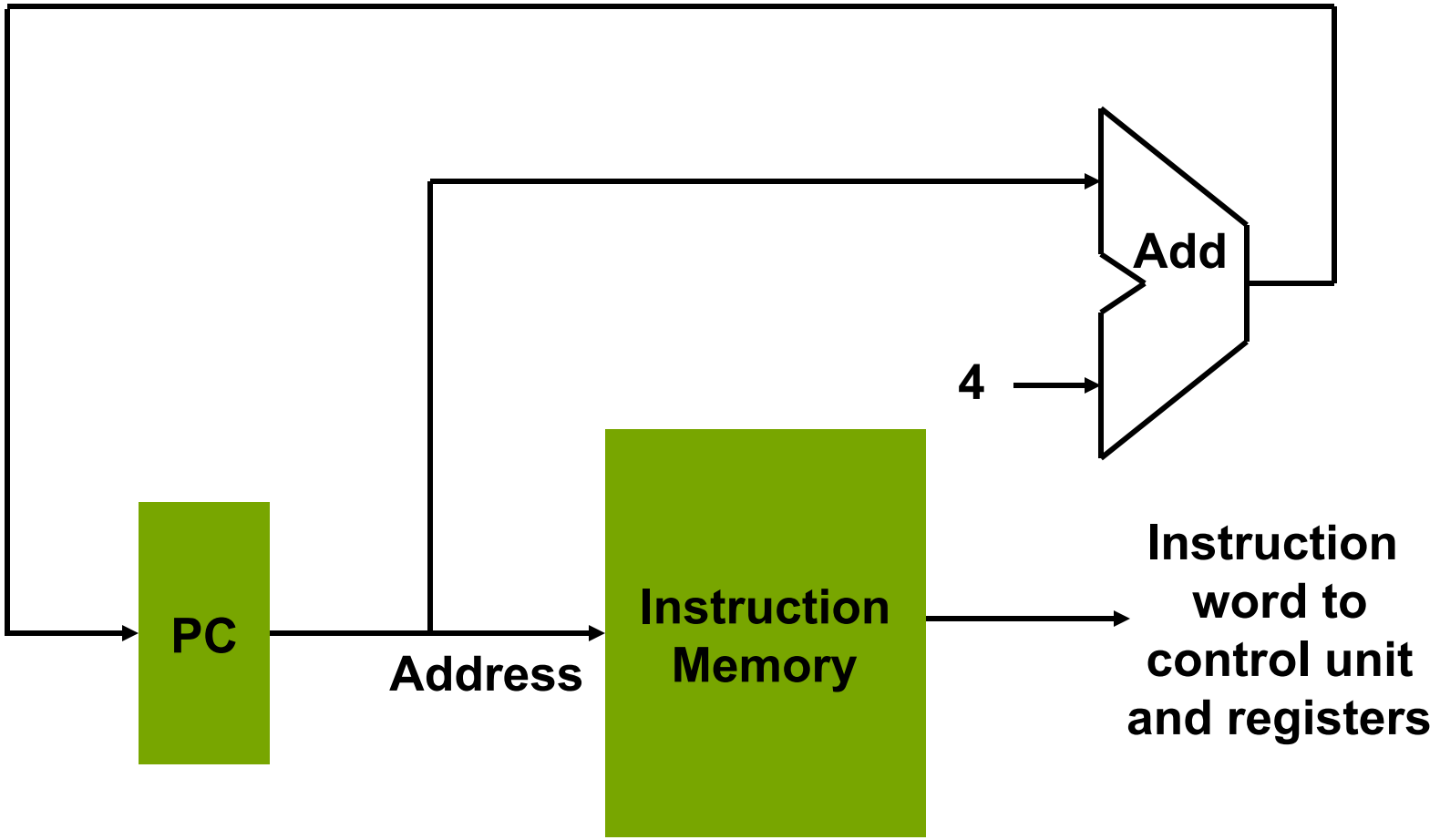
How Does It Run?



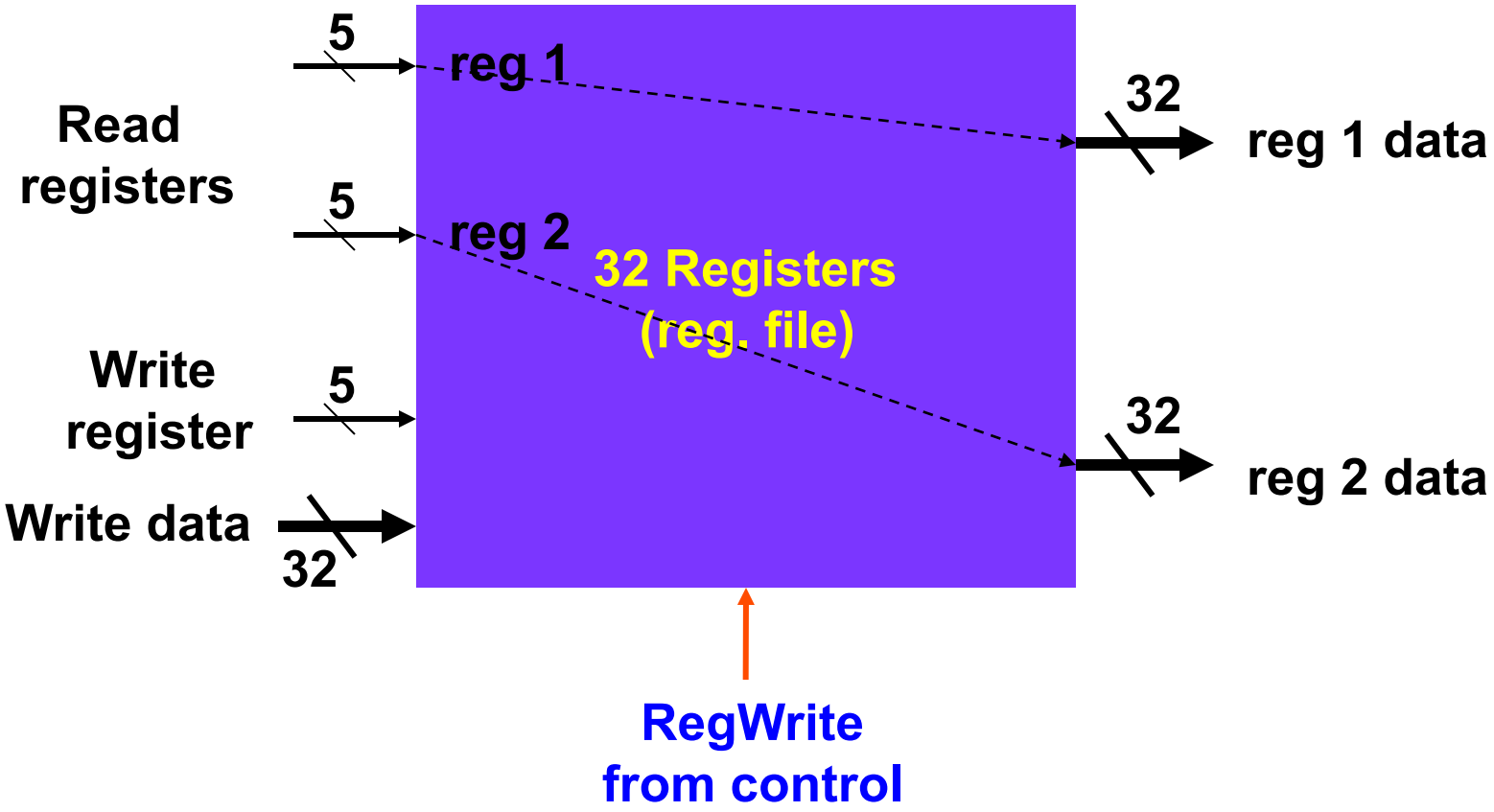
Datapath and Control

- Datapath: Memory, registers, adders, ALU, and communication buses. Each step (fetch, decode, execute) requires communication (data transfer) paths between memory, registers and ALU.
- Control: Datapath for each step is set up by control signals that set up dataflow directions on communication buses and select ALU and memory functions. Control signals are generated by a control unit consisting of one or more finite-state machines.

Datapath for Instruction Fetch

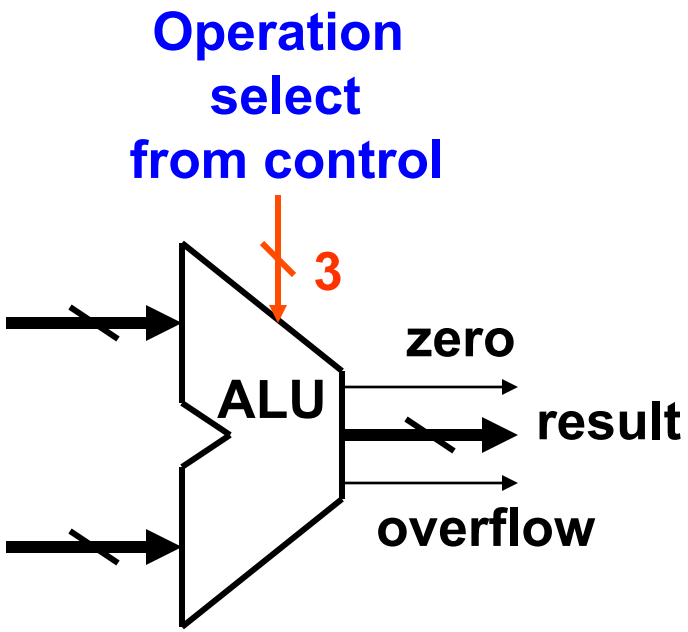


Register File: A Datapath Component



Multi-Operation ALU

Operation select	ALU function
000	AND
001	OR
010	Add
110	Subtract
111	Set on less than



zero = 1, when all bits of result are 0

Thank You