RISC Design

Introduction

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

EE-739: Processor Design

Lecture 16 (16 Feb 2015)

CADSL

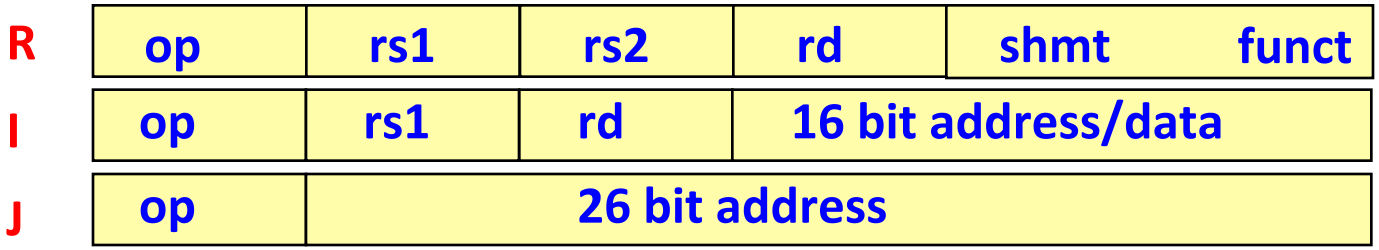
Example Instruction Set: MIPS Subset

MIPS Instruction – Subset

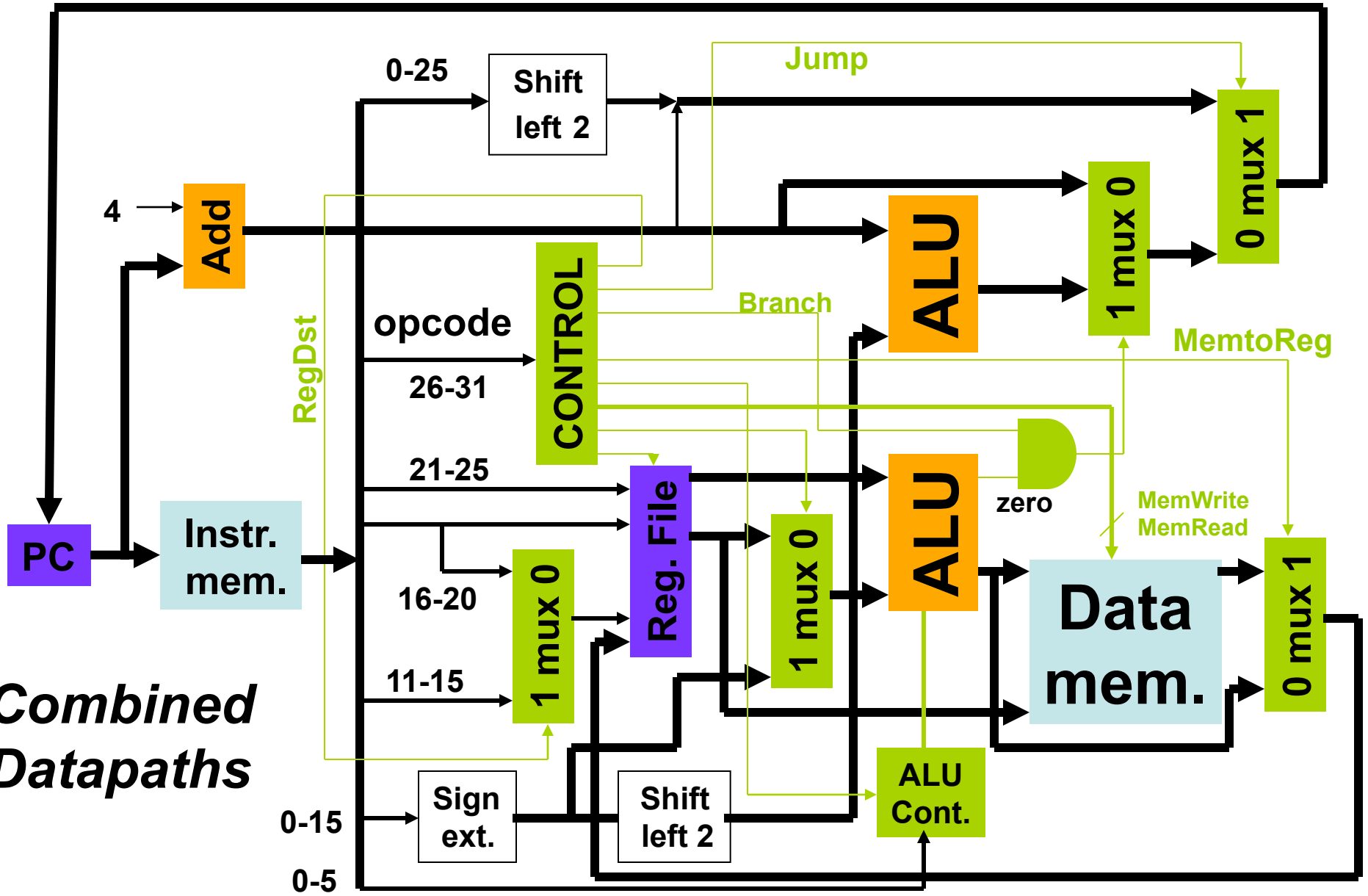
- Arithmetic and Logical Instructions
  - add, sub, or, and, slt
- Memory reference Instructions
  - lw, sw
- Branch
  - beq, j

Overview of MIPS

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



- rely on compiler to achieve performance


Combined Datapaths
0-25
Shift left 2
Jump
4
Add
ALU
1 mux 0
0 mux 1
RegDst
opcode
26-31
CONTROL
Branch
MemtoReg
21-25
Reg. File
ALU
zero
MemWrite
MemRead
PC
Instr. mem.
16-20
1 mux 0
1 mux 0
Data mem.
0 mux 1
11-15
0-15
Sign ext.
Shift left 2
ALU Cont.
0-5

How Long Does It Take?

- Assume control logic is fast and does not affect the critical timing. Major time delay components are ALU, memory read/write, and register read/write.
- Arithmetic-type (R-type)
  - Fetch (memory read) 2ns
  - Register read 1ns
  - ALU operation 2ns
  - Register write 1ns
  - Total 6ns

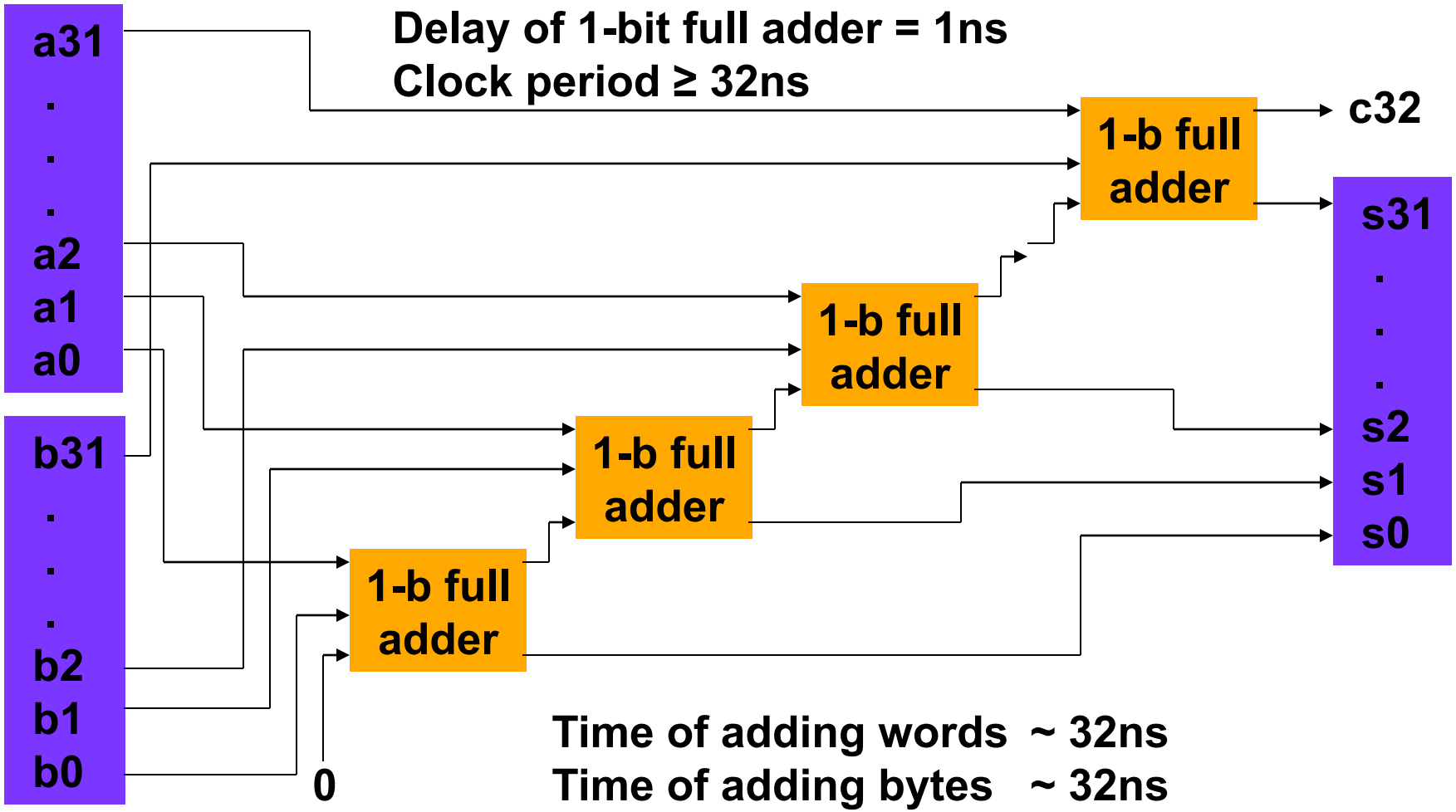
Execution Time

- ALU (R-type) 6ns
- Load word (I-type) 8ns
- Store word (I-type) 7ns
- Branch on equal (I-type) 5ns
- Jump (J-type)
  - Fetch (memory read) 2ns
  - Total 2ns

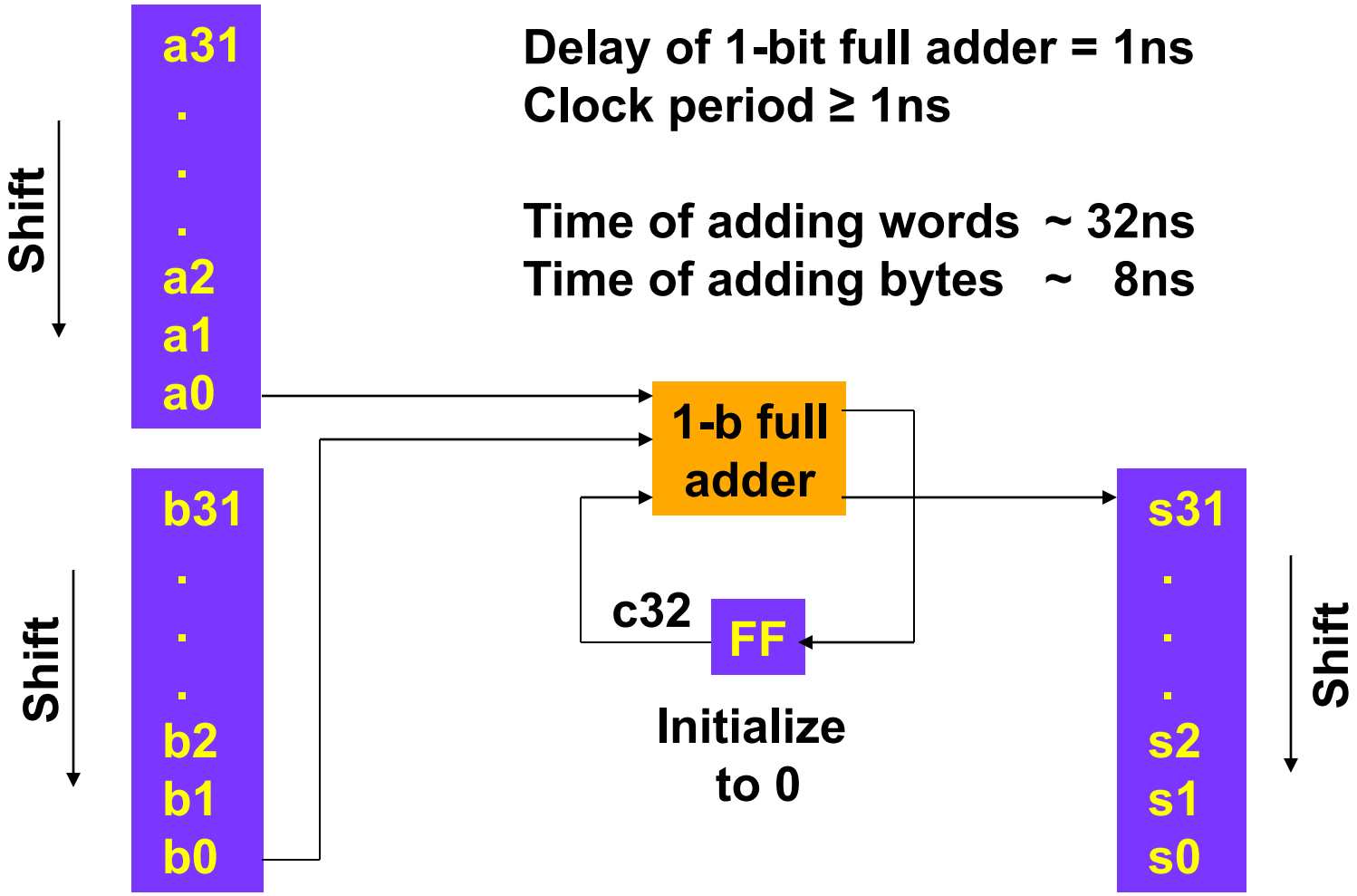
How Fast Can the Clock Be?

- If every instruction is executed in one clock cycle, then:
  - Clock period must be at least 8ns to perform the longest instruction, i.e., *lw*.
  - This is a single cycle machine (CPI = 1)
  - It is slower because many instructions take less than 8ns but are still allowed that much time.
- Method of speeding up: Use multicycle datapath.

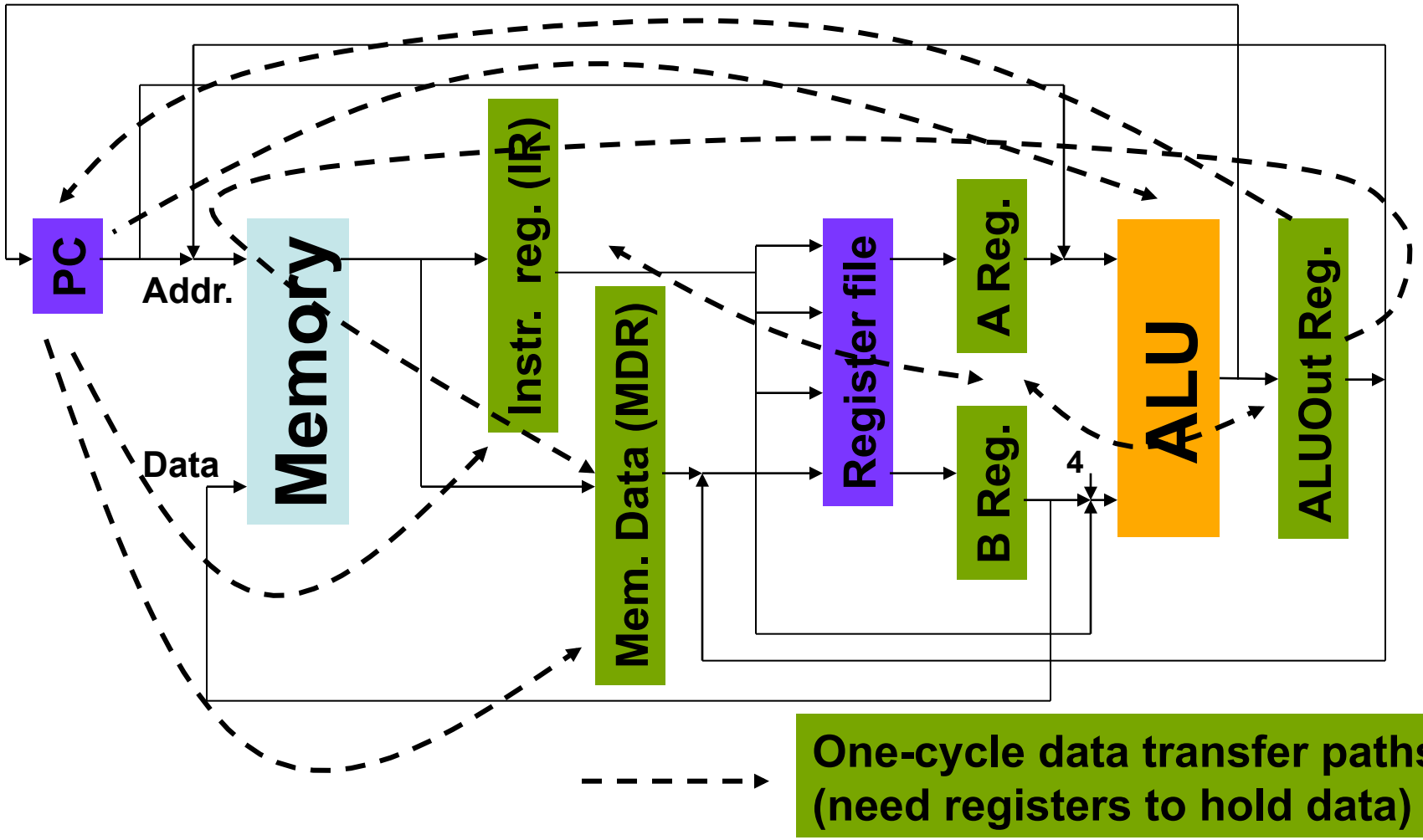
A Single Cycle Example



A Multicycle Implementation



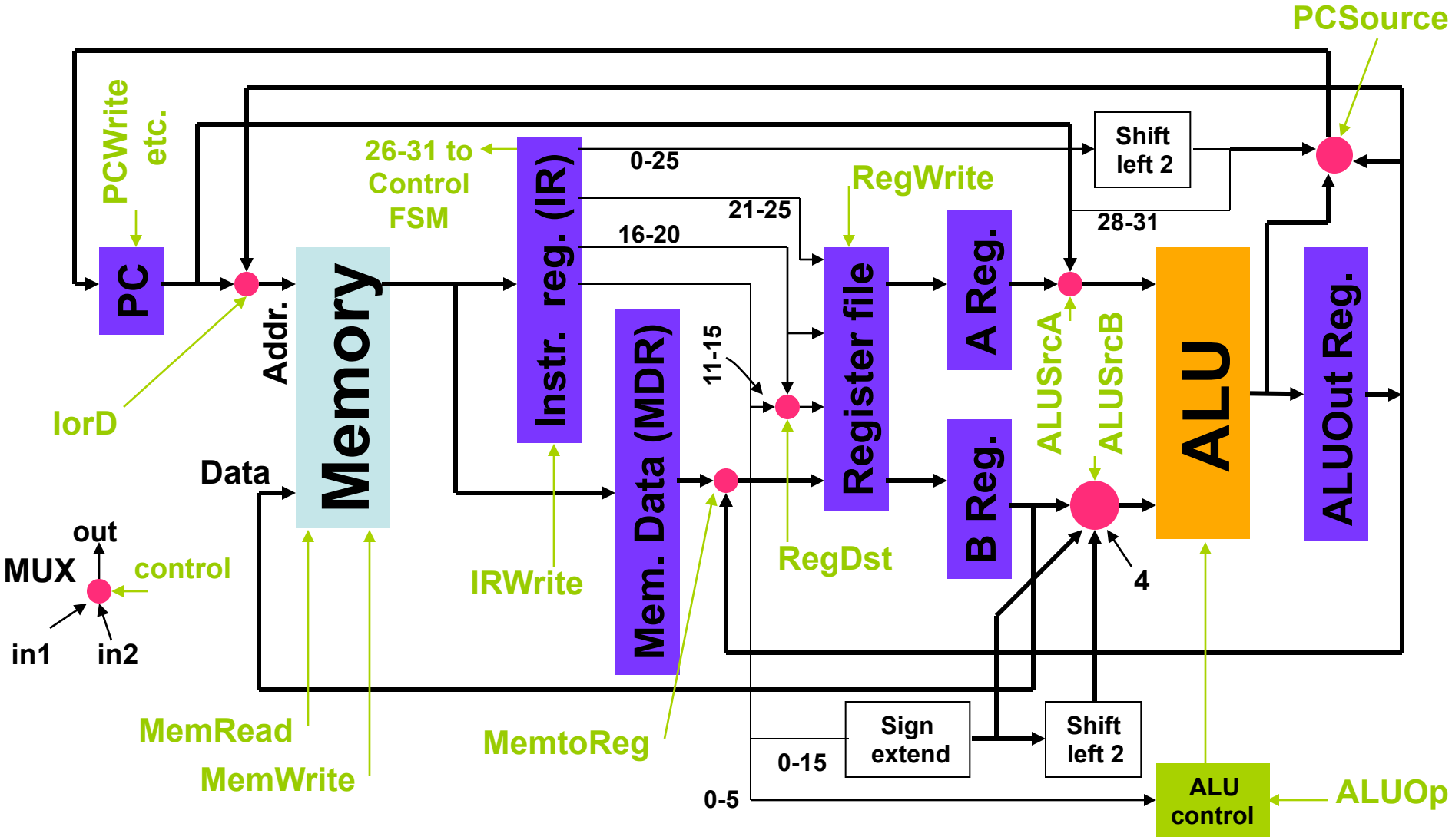
Multicycle Datapath



Multicycle Datapath Requirements

- Only one ALU, since it can be reused.
- Single memory for instructions and data.
- Five registers added:
  - Instruction register (IR)
  - Memory data register (MDR)
  - Three ALU registers, A and B for inputs and ALUOut for output

Multicycle Datapath



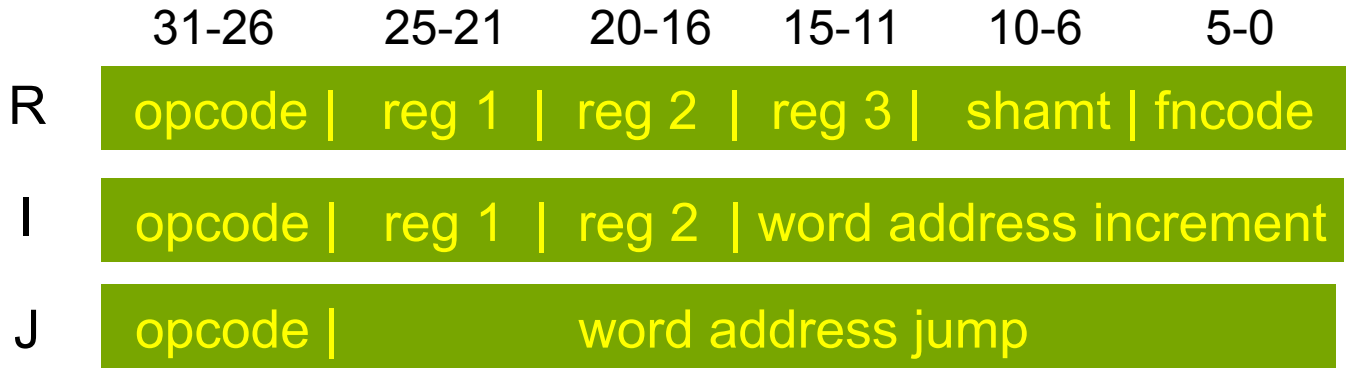
3 to 5 Cycles for an Instruction

Step	R-type (4 cycles)	Mem. Ref. (4 or 5 cycles)	Branch type (3 cycles)	J-type (3 cycles)
Instruction fetch	IR ← Memory[PC]; PC ← PC+4			
Instr. decode/ Reg. fetch	A ← Reg(IR[21-25]); B ← Reg(IR[16-20]) ALUOut ← PC + (sign extend IR[0-15]) << 2			
Execution, addr. Comp., branch & jump completion	**ALUOut ← A op B**	**ALUOut ← A+sign extend (IR[0-15])**	**If (A= =B) then PC←ALUOut**	**PC←PC[28-31] || (IR[0-25]<<2)**
Mem. Access or R-type completion	**Reg(IR[11-15]) ← ALUOut**	**MDR←M[ALUout] or M[ALUOut]←B**		
Memory read completion		**Reg(IR[16-20]) ← MDR**		

Cycle 1 of 5: Instruction Fetch (IF)

- Read instruction into IR, M[PC] → IR
  - Control signals used:
    - » IorD = 0 select PC
    - » MemRead = 1 read memory
    - » IRWrite = 1 write IR
- Increment PC, PC + 4 → PC
  - Control signals used:
    - » ALUSrcA = 0 select PC into ALU
    - » ALUSrcB = 01 select constant 4
    - » ALUOp = 00 ALU adds
    - » PCSource = 00 select ALU output
    - » PCWrite = 1 write PC

Cycle 2 of 5: Instruction Decode (ID)



- Control unit decodes instruction
- Datapath prepares for execution
  - R and I types, reg 1→ A reg, reg 2 → B reg
    - » No control signals needed
  - Branch type, compute branch address in ALUOut
    - » ALUSrcA = 0 select PC into ALU
    - » ALUSrcB = 11 Instr. Bits 0-15 shift 2 into ALU
    - » ALUOp = 00 ALU adds

Cycle 3 of 5: Execute (EX)

- R type: execute function on reg A and reg B, result in ALUOut
  - Control signals used:
    - » ALUSrcA = 1 A reg into ALU
    - » ALUsrcB = 00 B reg into ALU
    - » ALUOp = 10 instr. Bits 0-5 control ALU
- I type, lw or sw: compute memory address in ALUOut ← A reg + sign extend IR[0-15]
  - Control signals used:
    - » ALUSrcA = 1 A reg into ALU
    - » ALUSrcB = 10 Instr. Bits 0-15 into ALU
    - » ALUOp = 00 ALU adds

Cycle 3 of 5: Execute (EX)

- I type, beq: subtract reg A and reg B, write ALUOut to PC
  - Control signals used:
    - » **ALUSrcA = 1 A reg into ALU**
    - » **ALUsrcB = 00 B reg into ALU**
    - » **ALUOp = 01 ALU subtracts**
    - » **If zero = 1, PCSource = 01 ALUOut to PC**
    - » **If zero = 1, PCwriteCond = 1 write PC**
    - » **Instruction complete, go to IF**
- J type: write jump address to PC ← IR[0-25] shift 2 and four leading bits of PC
  - Control signals used:
    - » **PCSource = 10**
    - » **PCWrite = 1 write PC**
    - » **Instruction complete, go to IF**

Cycle 4 of 5: Reg Write/Memory

- R type, write destination register from ALUOut
  - Control signals used:
    - » **RegDst = 1 Instr. Bits 11-15 specify reg.**
    - » **MemtoReg = 0 ALUOut into reg.**
    - » **RegWrite = 1 write register**
    - » **Instruction complete, go to IF**
- I type, lw: read M[ALUOut] into MDR
  - Control signals used:
    - » **IorD = 1 select ALUOut into mem adr.**
    - » **MemRead = 1 read memory to MDR**
- I type, sw: write M[ALUOut] from B reg
  - Control signals used:
    - » **IorD = 1 select ALUOut into mem adr.**
    - » **MemWrite = 1 write memory**
    - » **Instruction complete, go to IF**

Cycle 5 of 5: Reg Write

- I type, lw: write MDR to reg[IR(16-20)]
  - Control signals used:
    - » RegDst = 0 instr. Bits 16-20 are write reg
    - » MemtoReg = 1 MDR to reg file write input
    - » RegWrite = 1 read memory to MDR
    - » **Instruction complete, go to IF**

For an alternative method of designing datapath, see
N. Tredennick, *Microprocessor Logic Design, the Flowchart Method*, Digital Press, 1987.

1-bit Control Signals

Signal name	Value = 0	Value =1
RegDst	Write reg. # = bit 16-20	Write reg. # = bit 11-15
RegWrite	No action	Write reg. ← Write data
ALUSrcA	First ALU Operand ← PC	First ALU Operand←Reg. A
MemRead	No action	Mem.Data Output←M[Addr.]
MemWrite	No action	M[Addr.]←Mem. Data Input
MemtoReg	Reg.File Write In←ALUOut	Reg.File Write In←MDR
IorD	Mem. Addr. ← PC	Mem. Addr. ← ALUOut
IRWrite	No action	IR ← Mem.Data Output
PCWrite	No action	PC is written
PCWriteCond	No action	PC is written if zero(ALU)=1

2-bit Control Signals

Signal name	Value	Action
ALUOp	00	ALU performs add
	01	ALU performs subtract
	10	Funct. field (0-5 bits of IR) determines ALU operation
ALUSrcB	00	Second input of ALU ← B reg.
	01	Second input of ALU ← 4 (constant)
	10	Second input of ALU ← 0-15 bits of IR sign ext. to 32b
	11	Second input of ALU ← 0-15 bits of IR sign ext. and left shift 2 bits
PCSource	00	ALU output (PC +4) sent to PC
	01	ALUOut (branch target addr.) sent to PC
	10	Jump address IR[0-25] shifted left 2 bits, concatenated with PC+4[28-31], sent to PC

Thank You