A Multiclock Design

A multiclock design allows datapath components to be used multiple times in the execution of an instruction, saving hardware. The design can be generalized later to a pipelined design for even greater speed.

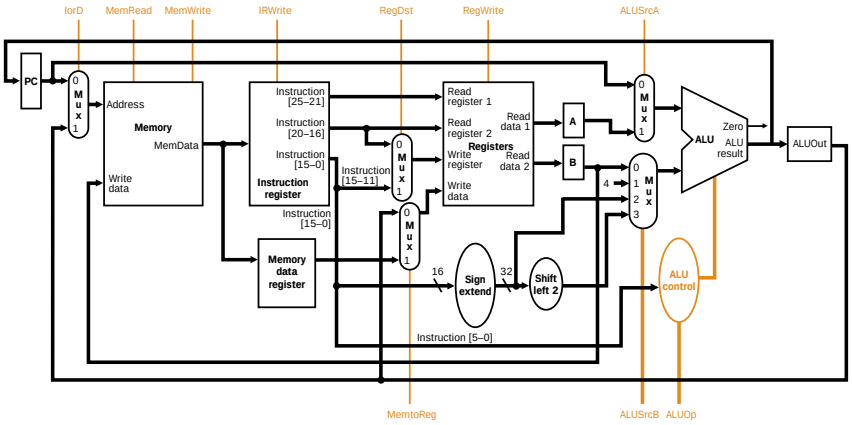
Five Cycle Instruction Execution

- Instruction Fetch
- Instruction decode and register fetch
- Execution, memory address computation or branch completion
- Memory access or R-type instruction completion
- Memory read completion

Goal: Separate logical operations so that the maximum number of instructions can make use of (variations of) each step

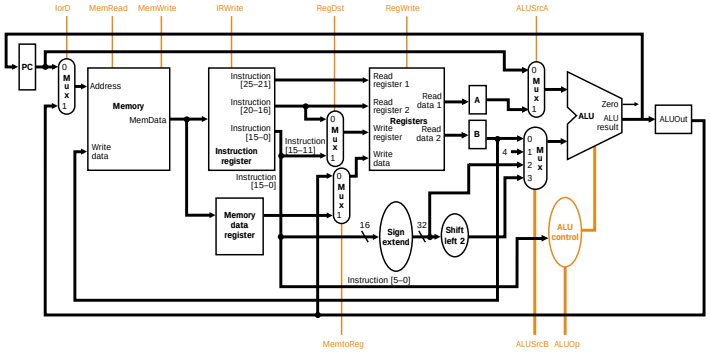
Extend Data Path

Add registers: IR, MDR, A, B, ALUOut



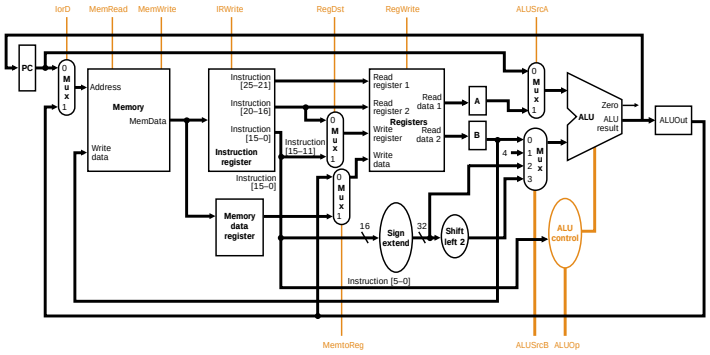
Control Signals

Signal	Effect Deasserted	Effect Asserted
MemRead	None	Contents at address read out
MemWrite	None	Contents at address replaced
ALUSelA	PC is first ALU operand	Reg is first ALU operand
RegDst	Reg Destination from rt	Reg Destination from rd
RegWrite	None	Contents of reg replaced
MemtoReg	Data to reg from ALU	Data to reg from memory
IorD	Mem Address from PC	Mem Addr from ALU
IRWrite	None	Value from memory to IR
PCWrite	None	PC is written
PCWriteCond	None	PC is written if Zero asserted



More Control Signals

Signal	Value	Effect
ALUSelB	00	Second ALU input from register rt
	01	Second ALU input is constant 4
	10	Second ALU input is sign extended 16 lsb of IR
	11	Second ALU input is sign extended 16 lsb of IR shifted
ALUOp	00	ALU performs addition
	01	ALU performs subtraction
	10	ALU operation comes from function code of instruction
PCSource	00	Output of ALU(PC+4) is sent to PC for writing
	01	Contents of ALUOut (branch target)are sent to PC for writing
	10	Jump target adddress is sent to PC for writing

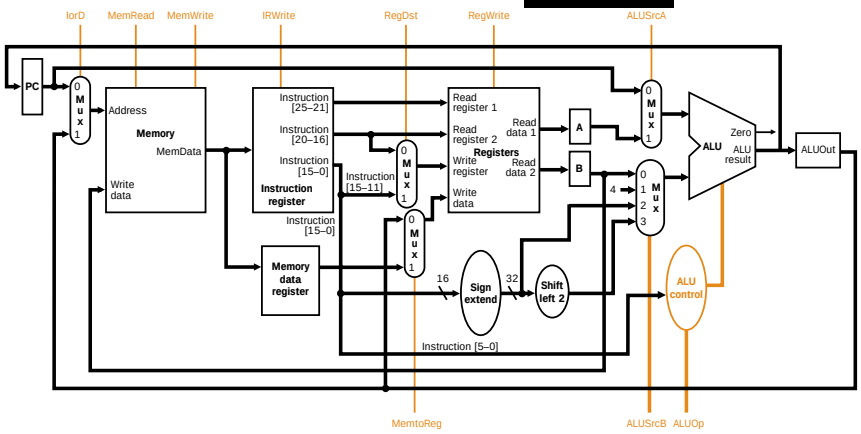


Instruction Cycle Descriptions

Step Name	Action for R-type	Action for Mem Ref	Branch
Instruction fetch	IR=Memory[PC] PC=PC+4		
Instruction decode/ register fetch	A=Reg[IR[25:21]] B=Reg[IR[20:16]] Target=PC+(xtnd(IR[15:0])<<2)		
Execution, addr comp or branch completion	ALUout=A op B	ALUout=A+ xtnd(IR[15:0]])	if (A==B) then PC=Target
Memory access or R-type completion	Reg[IR[15:11]= ALUout	memdata=ALUout or Mem[ALUout]=B	
Write back		Reg[IR[20:16]]= memdata	

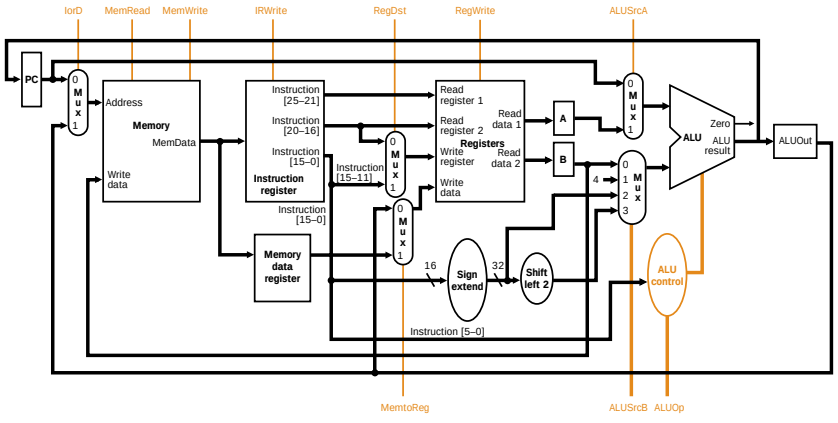
Instruction Fetch

MemRead
IRWrite
IorD = 0
ALUSelB=01
ALUSelA=0
ALUOp=00



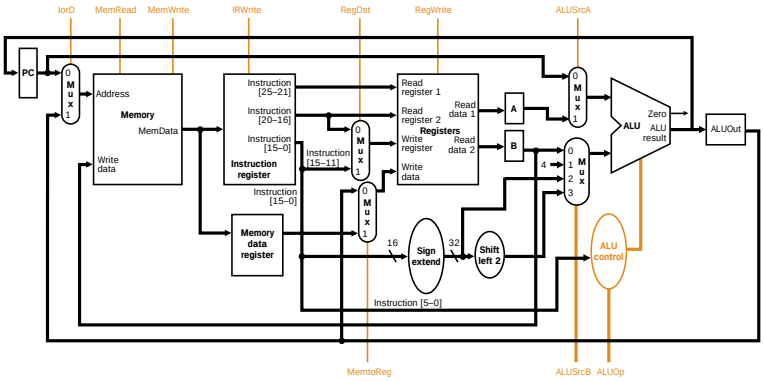
Instruction Decode & Register Fetch

ALUSelB=11
ALUSelA=0
ALUOp=00



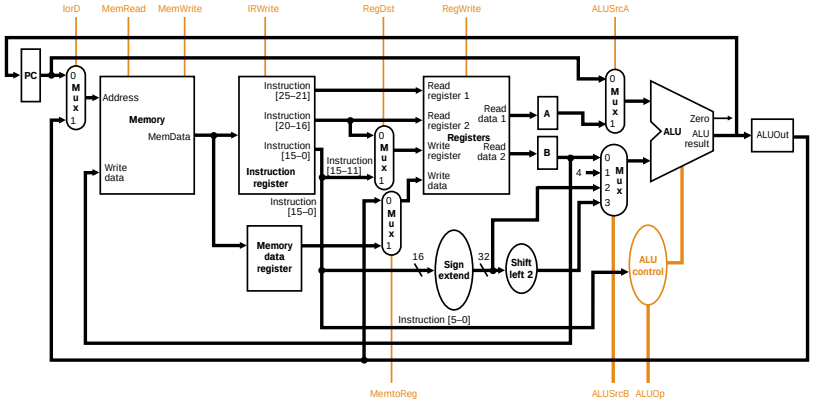
Execution ...

- Exeuction: ALUOut = A op B
- Mem Addressing: ALUOut = A + xtnd(IR[15:0])
- Branch Completion: if (A==B) PC = Target



Mem Access or R-type Complete

- Memory Ref: MemData = Memory[ALUOut] or
  Memory[ALUOut] = B
- R-type Completion: Reg[IR[15:11]]=ALUOut



Write Back

- Register[IR[20-16]] = MemDataReg