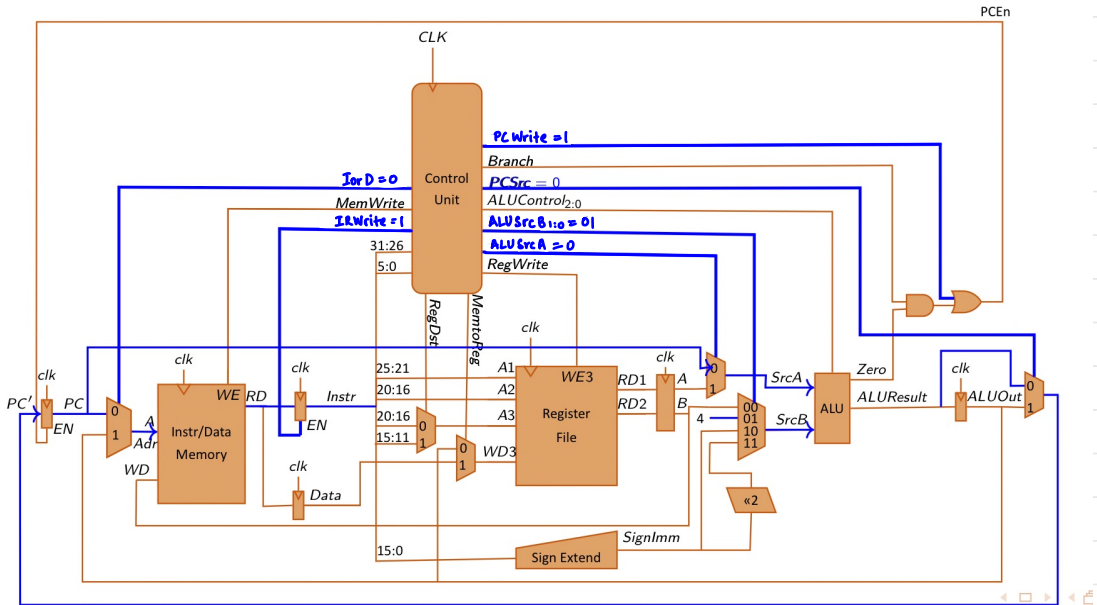# R-Type Instruction

- 4 clock cycles

Clock cycle #1

- fetch instruction

PCEn

CLK

Control Unit

PCWrite = 1
Branch
$PCSrc = 0$
$ALUControl_{2:0}$
$ALUSrcB_{1:0} = 01$
$ALUSrcA = 0$
RegWrite

IorD = 0
MemWrite
IRWrite = 1

31:26
5:0

RegDst
MemtoReg

clk

PC' PC
clk   clk

EN

Instr/Data Memory
WE RD
A
Adr
WD

clk

Instr

clk
EN

Data

25:21
20:16
20:16
15:11

A1
A2
A3
WD3

WE3
Register File
RD1
RD2

clk

A
B

SrcA
ALU
SrcB
4

00
01
10
11

Zero
ALUResult

clk
ALUOut

15:0

Sign Extend

SignImm

«2

So: fetch

reset →

IorD = 0
ALUSrc A = 0
ALUSrc B = 01
ALUOp = 00
PC Src = 0
IRWrite
PCWrite

- decode state

PCEn

CLK

Control Unit

PCWrite
Branch
PCSrc
$ALUControl_{2:0}$
$ALUSrcB_{1:0}$
ALUSrcA
RegWrite

IorD
MemWrite
IRWrite
31:26
5:0

RegDst
MemtoReg

clk

PC' PC

clk

WE RD

EN

A
Adr
WD

Instr/Data
Memory

clk

clk

Instr

EN

clk

Data

25:21
20:16
20:16
15:11

15:0

0
1

0
1

WD3

A1
A2
A3

WE3

Register
File

RD1
RD2

clk

A
B

0
1

SrcA

4

00
01
10
11

SrcB

«2

ALU

Zero
ALUResult

clk

ALUOut

0
1

Sign Extend

SignImm

S0: fetch

S1: decode

reset

IorD = 0
ALUSrcA = 0
ALUSrcB = 01
ALUOp = 00
PCSrc = 0
IRWrite
PCWrite

CLK

PCEn

Control Unit

PCWrite
Branch
PCSrc
ALUControl$_{2:0}$
ALUSrc B$_{1:0}$ = 00
ALUSrc A = 1
RegWrite

IorD
MemWrite
IRWrite
31:26
5:0

RegDst
MemtoReg

clk

PC' PC

Instr/Data Memory

Register File

ALU

A1
A2
A3

WE3
RD1
RD2

A
B

SrcA
SrcB

Zero
ALUResult

ALUOut

Sign Extend

SignImm

«2

S0: fetch

IorD=0
ALUSrcA=0
ALUSrcB=01
ALUOp=00
PCSrc=0
IRWrite
PCWrite

reset

S1: decode

Op=R-Type

S6: execute

ALUSrcA=1
ALUSrcB=00
ALUOp=10

add/
sub/
and/
or/
slt

Clock cycle #4

S0: fetch

IorD = 0
ALUSrcA = 0
ALUSrcB = 01
ALUOp = 00
PCSrc = 0
IRWrite
PCWrite

reset →

S1: decode

Op = R-Type →

S6: execute

ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10

S7: ALU writeback

RegDst = 1
MemtoReg = 0
RegWrite

Branch if Equal Instruction

Clock cycle #1

So: fetch

reset →

IorD=0
ALUSrcA=0
ALUSrcB=01
ALUOp=00
PCSrc=0
IRWrite
PCWrite

- decode
- compute branch target address

PCEn

CLK

PCWrite
Branch
PCSrc
ALUControl$_{2:0}$
ALUSrcB$_{1:0}$ = 11
ALUSrc A = 0
RegWrite

IorD
MemWrite
IRWrite
31:26
5:0

Control
Unit

clk

PC' PC

Instr/Data
Memory

WE RD
Instr

Register
File

A1
A2
A3
WD3

RD1 A
RD2 B

SrcA
SrcB

ALU

Zero
ALUResult

ALUOut

Sign Extend

SignImm

«2

S0: fetch

IorD = 0
ALUSrc A = 0
ALUSrc B = 01
ALUOp = 00
PCSrc = 0
IRWrite
PCWrite

reset

S1: decode

ALUSrc A = 0
ALUSrc B = 11
ALUOp = 00

Clock cycle #3

CLK

Control Unit

- IorD
- MemWrite
- IRWrite
- 31:26
- 5:0

- PCWrite
- **Branch = 1**
- **PCSrc = 0**
- $ALUControl_{2:0}$
- **$ALUSrcB_{1:0} = 00$**
- **$ALUSrcA = 1$**
- RegWrite

PCEn

Instr/Data Memory — WE RD, A Adr, WD, Data

Register File — A1, A2, A3, WE3, RD1, RD2, WD3

ALU — SrcA, SrcB, Zero, ALUResult, ALUOut

Sign Extend — 15:0, SignImm

RegDst, MemtoReg

«2

S0: fetch

IorD = 0
ALUSrcA = 0
ALUSrc B = 01
ALUOp = 00
PC Src = 0
IRWrite
PCWrite

reset →

S1: decode

ALUSrcA = 0
ALUSrc B = 11
ALUOp = 00

Op = BEQ →

S8: Branch

ALUSrcA = 1
ALUSrc B = 00
ALUOp = 01
PCSrc = 1
Branch

Control FSM

- For each instruction, FSMs

same

LW

S0: Fetch
IorD = 0
AluSrcA = 0
AluSrcB = 01
AluOp = 00
PCSrc = 0
IRWrite
PCWrite

Reset →

S1: Decode

Op = LW
or
Op = SW

S2: MemAdr
AluSrcA = 1
AluSrcB = 10
AluOp = 00

Op = LW

S3: MemRead
IorD = 1

same

S4: Mem Writeback
RegDst = 0
MemtoReg = 1
RegWrite

SW

S0: Fetch
IorD = 0
AluSrcA = 0
AluSrcB = 01
AluOp = 00
PCSrc = 0
IRWrite
PCWrite

Reset →

S1: Decode

Op = LW
or
Op = SW

S2: MemAdr
AluSrcA = 1
AluSrcB = 10
AluOp = 00

Op = SW

S5: MemWrite
IorD = 1
MemWrite

R-Type

S0: Fetch
IorD = 0
AluSrcA = 0
AluSrcB = 01
AluOp = 00
PCSrc = 0
IRWrite
PCWrite

Reset →

S1: Decode

Op = R-type

S6: Execute
AluSrcA = 1
AluSrcB = 00
AluOp = 10

S7: ALU Writeback
RegDst = 1
MemtoReg = 0
RegWrite

**same;
LW, SW &
R-Type
don't care
about control
signals;
all 4 can be
identical**

BEQ

S0: Fetch
IorD = 0
AluSrcA = 0
AluSrcB = 01
AluOp = 00
PCSrc = 0
IRWrite
PCWrite

Reset →

S1: Decode
AluSrcA = 0
AluSrcB = 11
AluOp = 00

Op = BEQ

S8: Branch
AluSrcA = 1
AluSrcB = 00
AluOp = 01
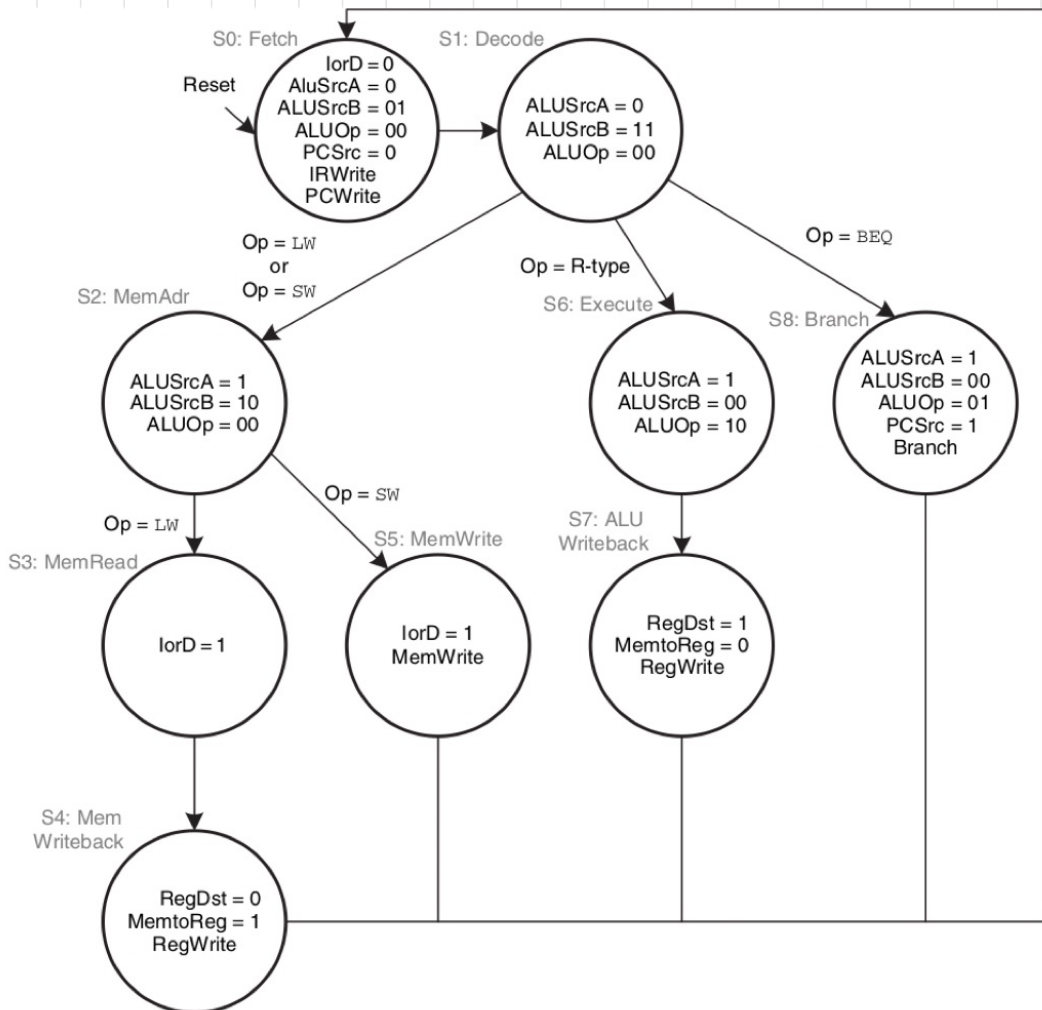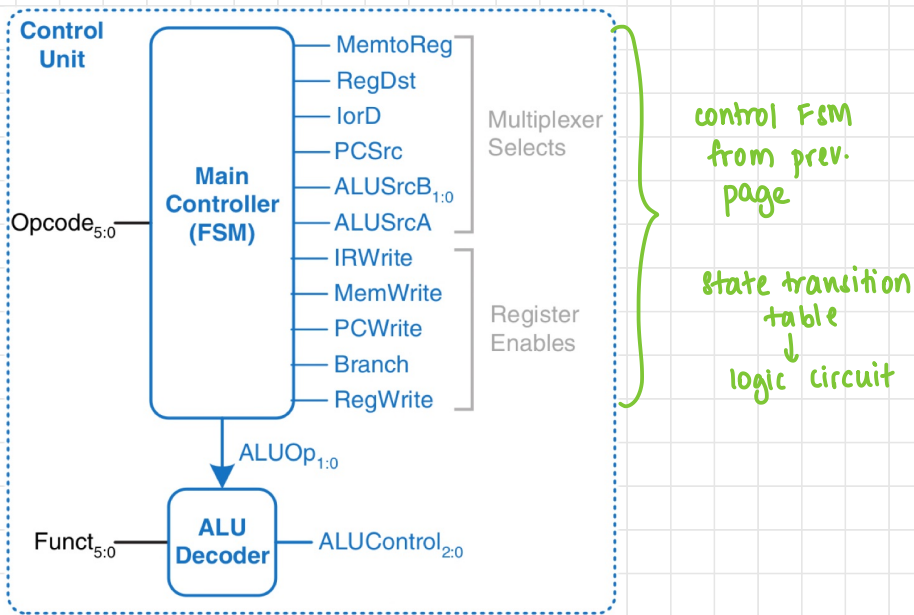PCSrc = 1
Branch

- Each final state connects back to first state (start state) to start next cycle

- First 2 states made common for all instructions, then branching for each instruction

Control FSM for Multi-Cycle Datapath

S0: Fetch
IorD = 0
AluSrcA = 0
ALUSrcB = 01
ALUOp = 00
PCSrc = 0
IRWrite
PCWrite

Reset

S1: Decode
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00

Op = LW
or
Op = SW

Op = R-type

Op = BEQ

S2: MemAdr
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00

S6: Execute
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10

S8: Branch
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCSrc = 1
Branch

Op = LW

Op = SW

S3: MemRead
IorD = 1

S5: MemWrite
IorD = 1
MemWrite

S7: ALU Writeback
RegDst = 1
MemtoReg = 0
RegWrite

S4: Mem Writeback
RegDst = 0
MemtoReg = 1
RegWrite

Control Unit

Main Controller (FSM) outputs:

Multiplexer Selects:
- MemtoReg
- RegDst
- IorD
- PCSrc
- $ALUSrcB_{1:0}$
- ALUSrcA

Register Enables:
- IRWrite
- MemWrite
- PCWrite
- Branch
- RegWrite

Inputs: $Opcode_{5:0}$

$ALUOp_{1:0}$

ALU Decoder — inputs $Funct_{5:0}$, output $ALUControl_{2:0}$

control FSM from prev. page

state transition table
↓
logic circuit

ALU Decoder

ALUOp	Funct	ALUControl
00	X	010 (add)
X1	X	110 (subtract)
1X	100000 (add)	010 (add)
1X	100010 (sub)	110 (subtract)
1X	100100 (and)	000 (and)
1X	100101 (or)	001 (or)
1X	101010 (slt)	111 (set less than)