Project #2 Assignment

03.06.2021

Project #2 Assignment

20 instructions

- R-format (6) : balrz, brn, brz, sll, sllv, srlv
- I-format (11) : andi, nori, xori, bgez, bgezal, bgtz, bltz, jalm, jm, jsp, jspal
- J-format (3) :balz, bz, jal

especially, jump and branches

Project #2 Assignment

You must

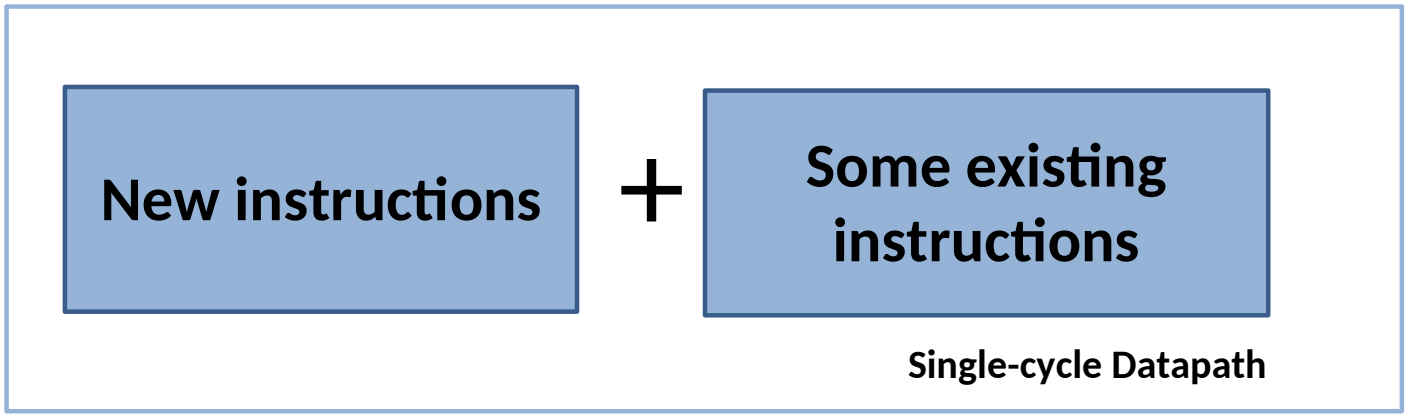
1) determine which instructions are actual in the current ISA.

2) determine which instructions are already implemented in the ‘MIPS-lite’ processor.

3) design the revised single-cycle datapath and revised control units which make a processor that executes all your instructions.

4) implement “your” single cycle processor in Verilog HDL.

5) prepare a simulation for the demo.



- Your Goal&Task
  - extend ***MIPS ISA*** (instruction set architecture)
  - modify datapath and control
  - ***single-cycle*** processor implementation

We already have in the course book

- page 279
- MIPS Reference Data Card
- Appendix B

3 steps

1) RTL

2) Changes to the Datapath

3) Design the Control table(s)

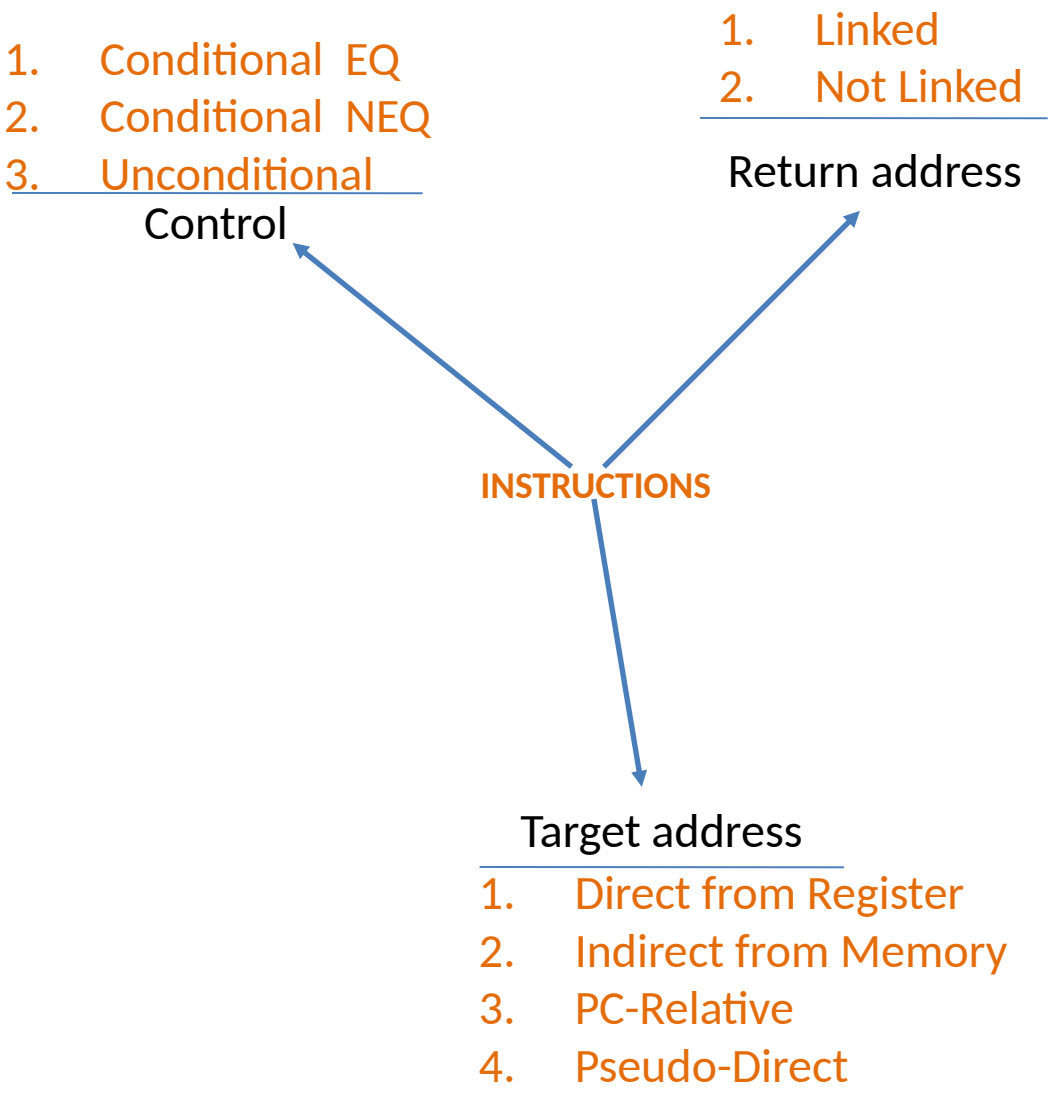
3 aspects of control Instructions

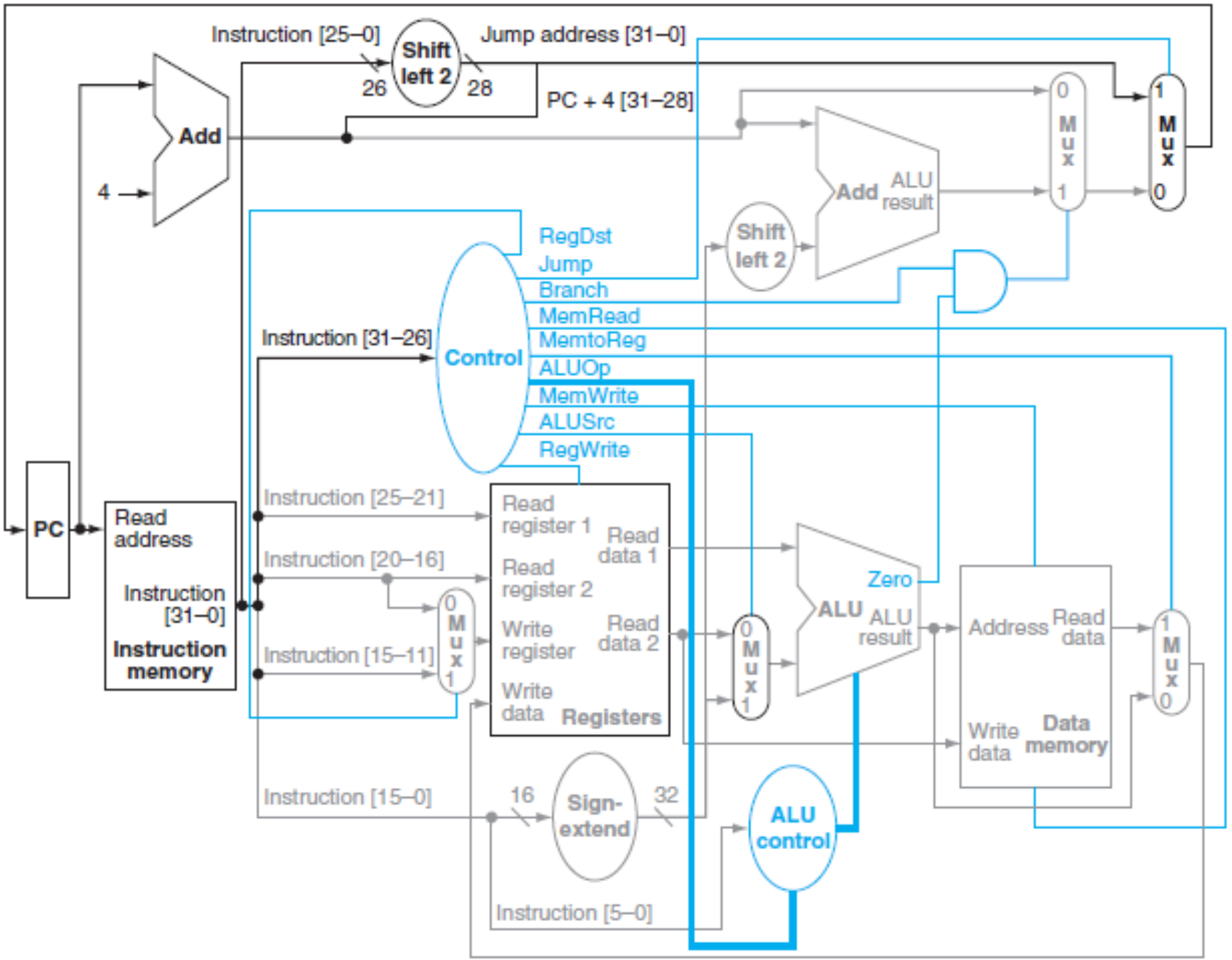
1. the change of control is conditional or unconditional

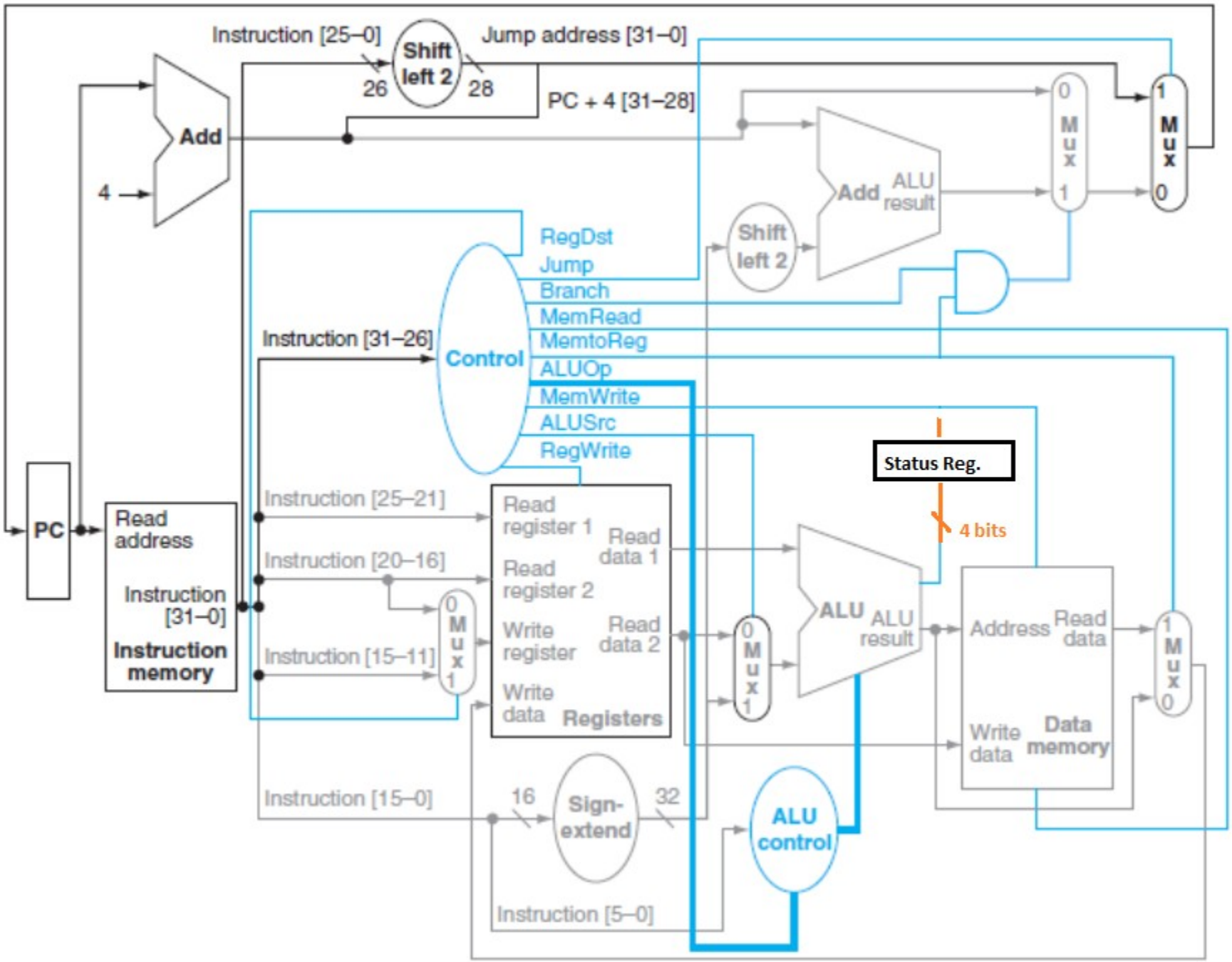
2. a return address link is stored or not

3. which option for the target address is chosen

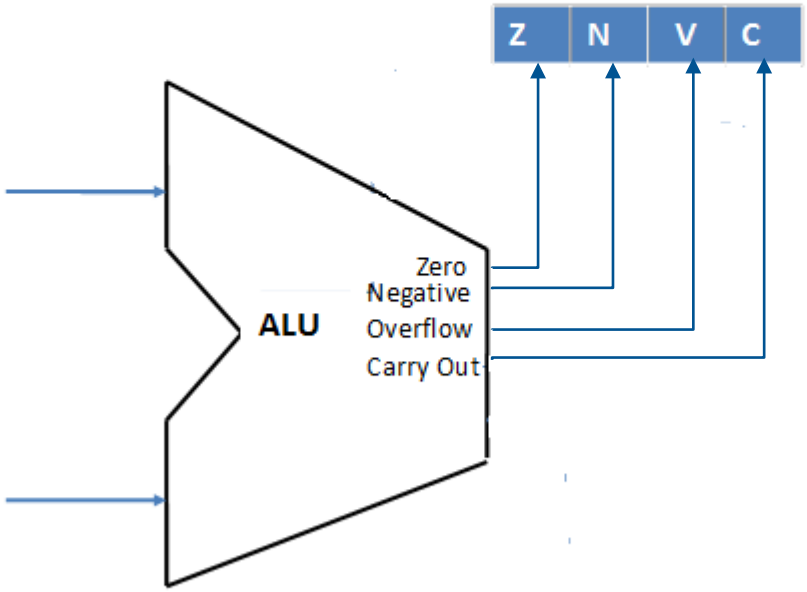
3 aspects of control Instructions




Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
0
M
u
x
1
1
M
u
x
0
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ALU
Zero
ALU result
Address
Read data
Write data
Data memory
16
Sign-extend
32
ALU control


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
0
M
u
x
1
1
M
u
x
0
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
Status Reg.
4 bits
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]

Status register



Conditional change of control (branch)

if Status [Z] = 1

instruction	syntax	branch address	link address
balrz	**balrz $rs, $rd**	*PC ← R[rs]*	*R[rd] ← PC + 4*
brz	**brz $rs**	*PC ← R[rs]*	---
bz	**bz Target**	Pseudo-direct address *PC ← BAddr* *PC ← PC[31:28]||[25:0]||00*	---
balz	**balz Target**	Pseudo-direct address *PC ← Baddr*	*R[31] ← PC + 4*

Unconditional change of control (jump)

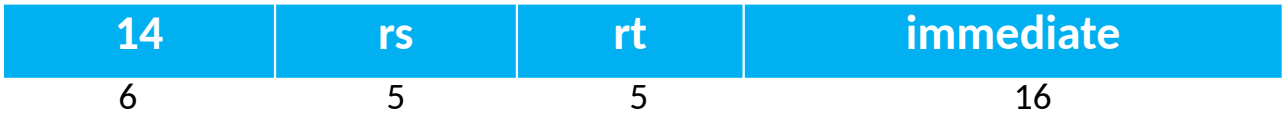
instruction	syntax	jump address	link address
jm	**jm imm16($rs)**	*PC* ← *M*	---
jalm	**jalm $rt, imm16($rs)**	*PC* ← *M*	*R[rt]* ← *PC + 4*
jsp	**jsp**	*PC* ← *R[29]*	---
jspal	**jspal**	*PC* ← *R[29]*	*M[R[29]]* ← *PC + 4*

- R-format

  balrz rs, rd

- I-format

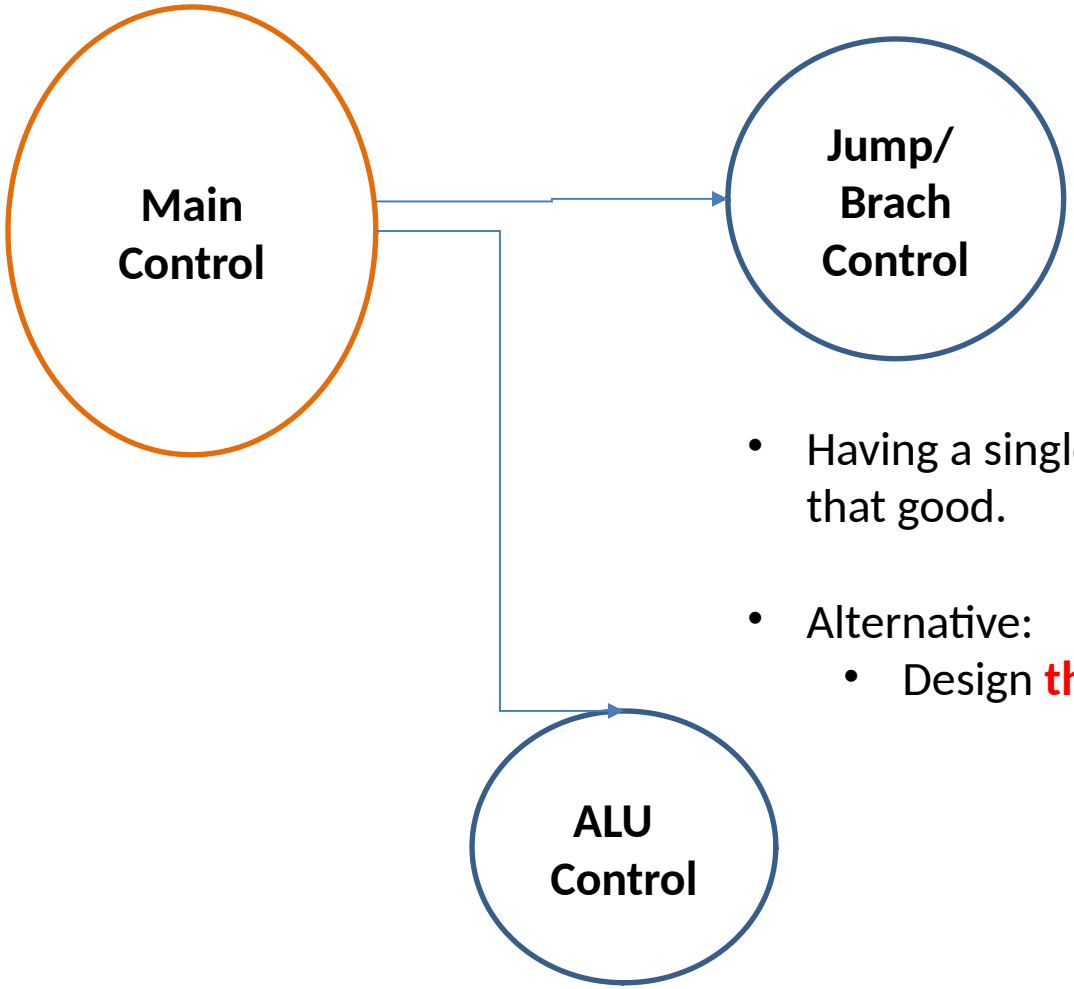
  xori rs, rt, Imm



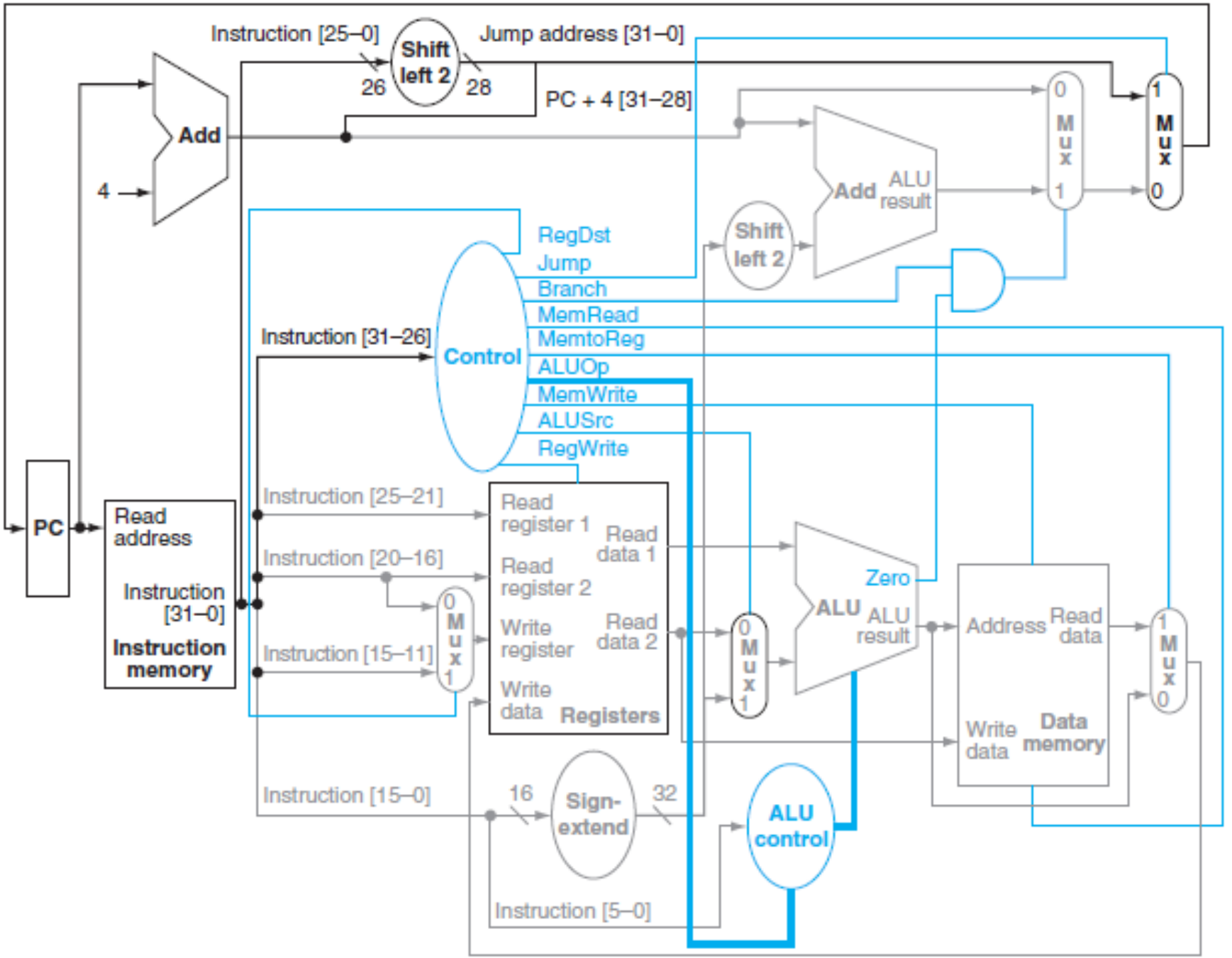
- J-format

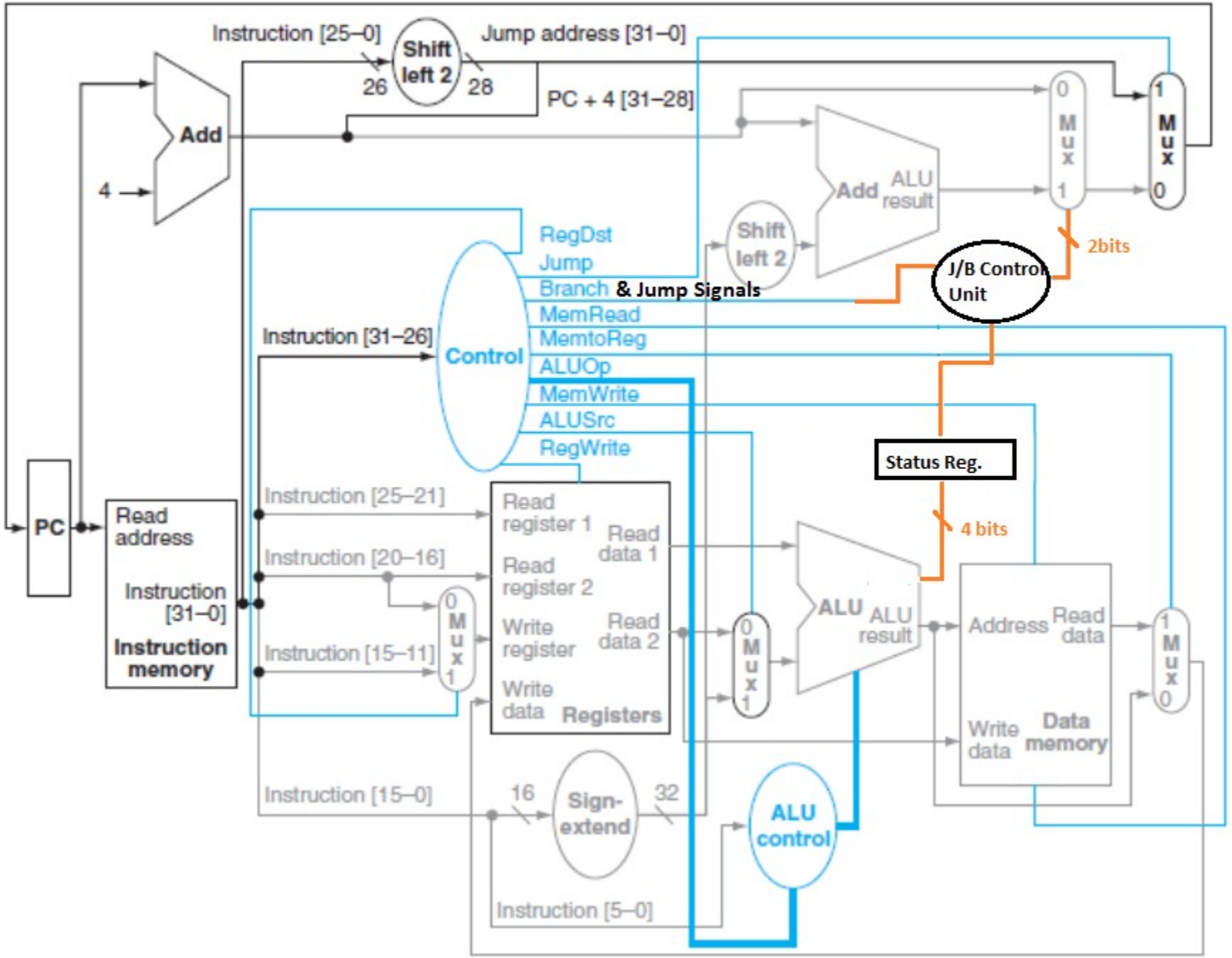
  jal target

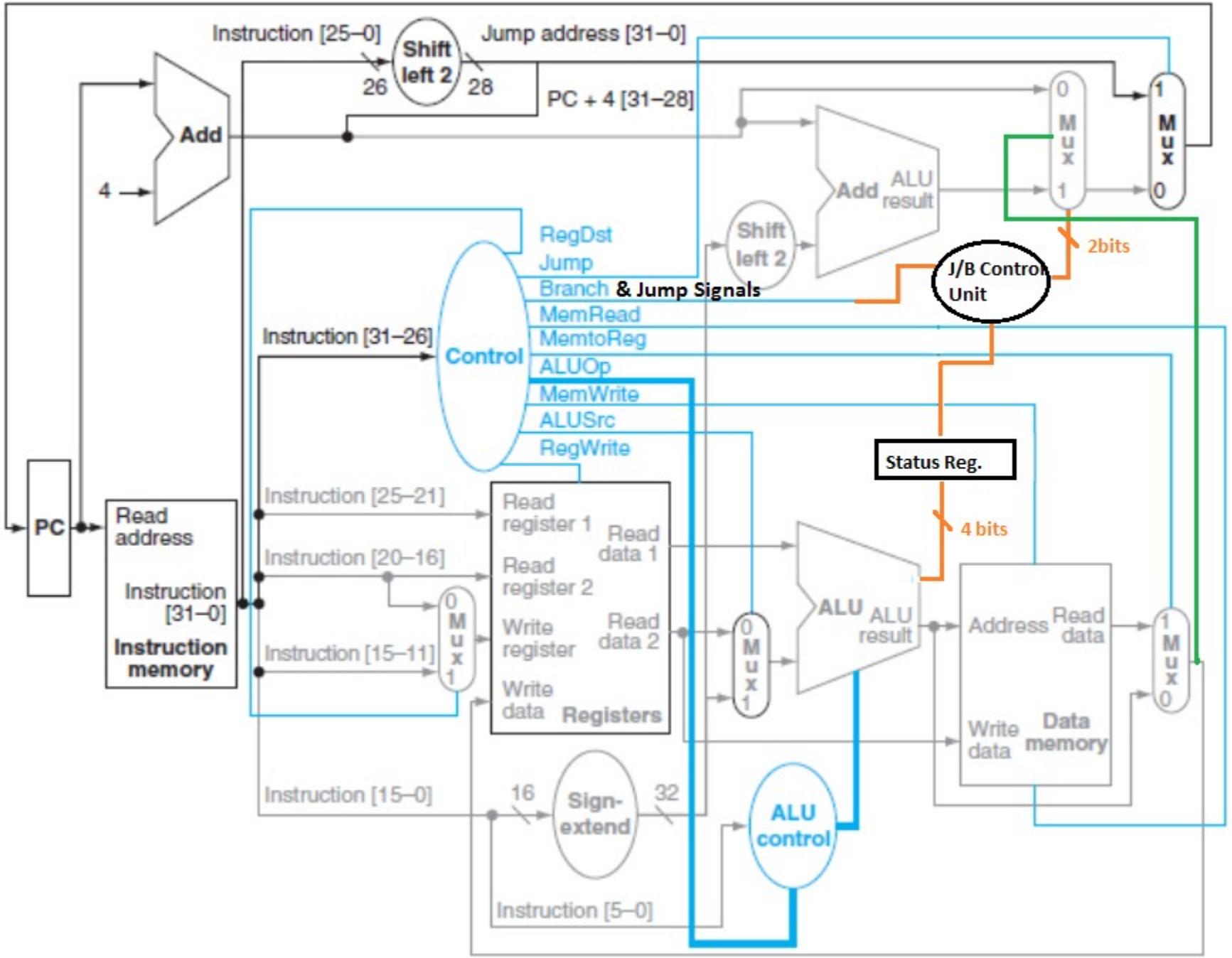
Control Units

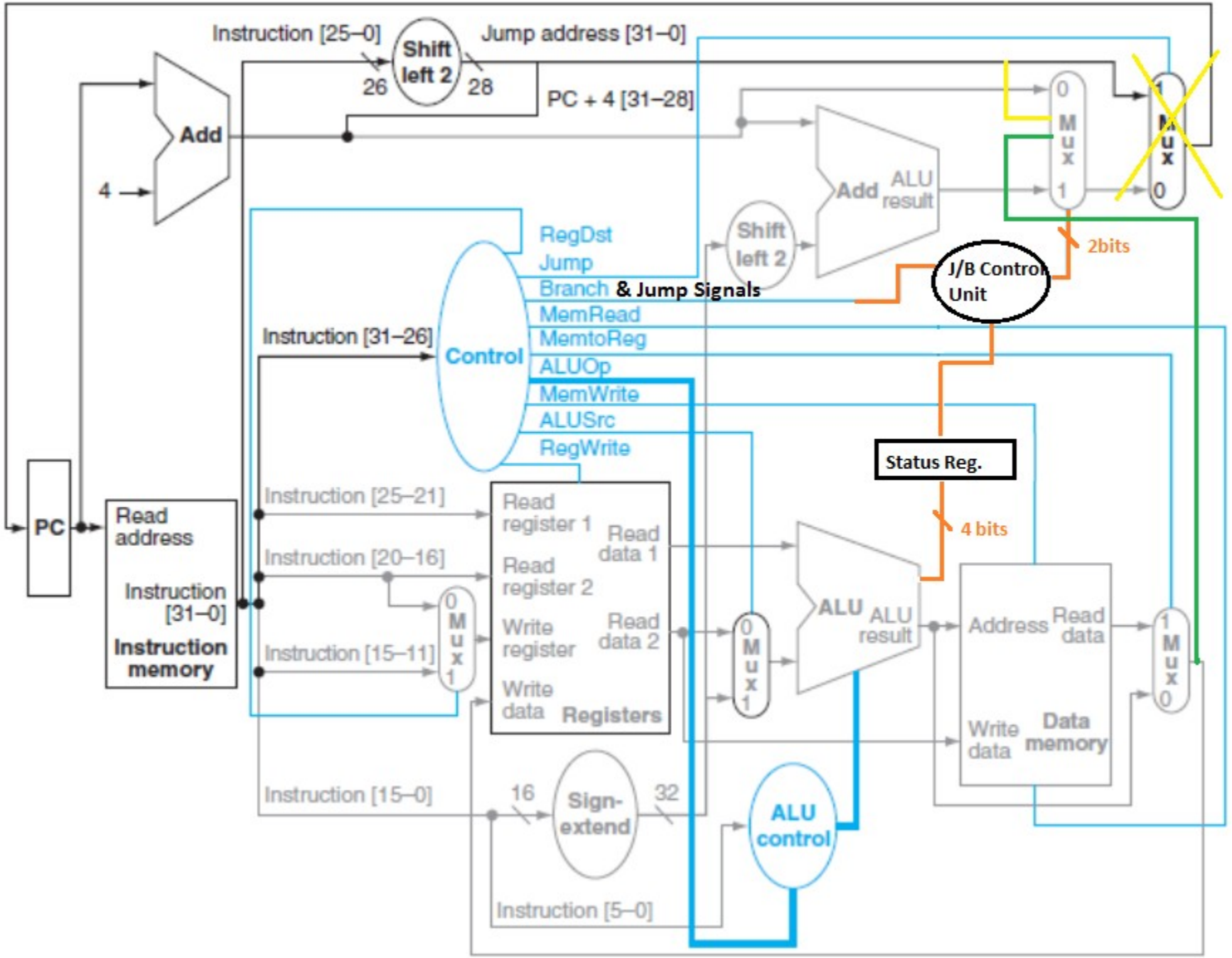


- Having a single Control Table is not that good.
- Alternative:
  - Design **three** control tables


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
0
M
u
x
1
1
M
u
x
0
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
Add
4
PC + 4 [31–28]
0 M u x 1
1 M u x 0
Add
ALU result
Shift left 2
2bits
J/B Control Unit
RegDst
Jump
Branch & Jump Signals
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
Status Reg.
4 bits
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
Add
4
PC + 4 [31–28]
0 M u x 1
1 M u x 0
Add
ALU result
Shift left 2
2bits
J/B Control Unit
RegDst
Jump
Branch & Jump Signals
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
Status Reg.
4 bits
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
0
M
u
x
1
Add
ALU result
Shift left 2
RegDst
Jump
Branch
& Jump Signals
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
J/B Control Unit
2bits
Status Reg.
4 bits
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]