Adding Support for jal to Single Cycle Datapath

(For More Practice Exercise 5.20)

- **The MIPS jump and link instruction, jal is used to support procedure calls by jumping to jump address (similar to j) and saving the address of the following instruction PC+4 in register $ra ($31)**

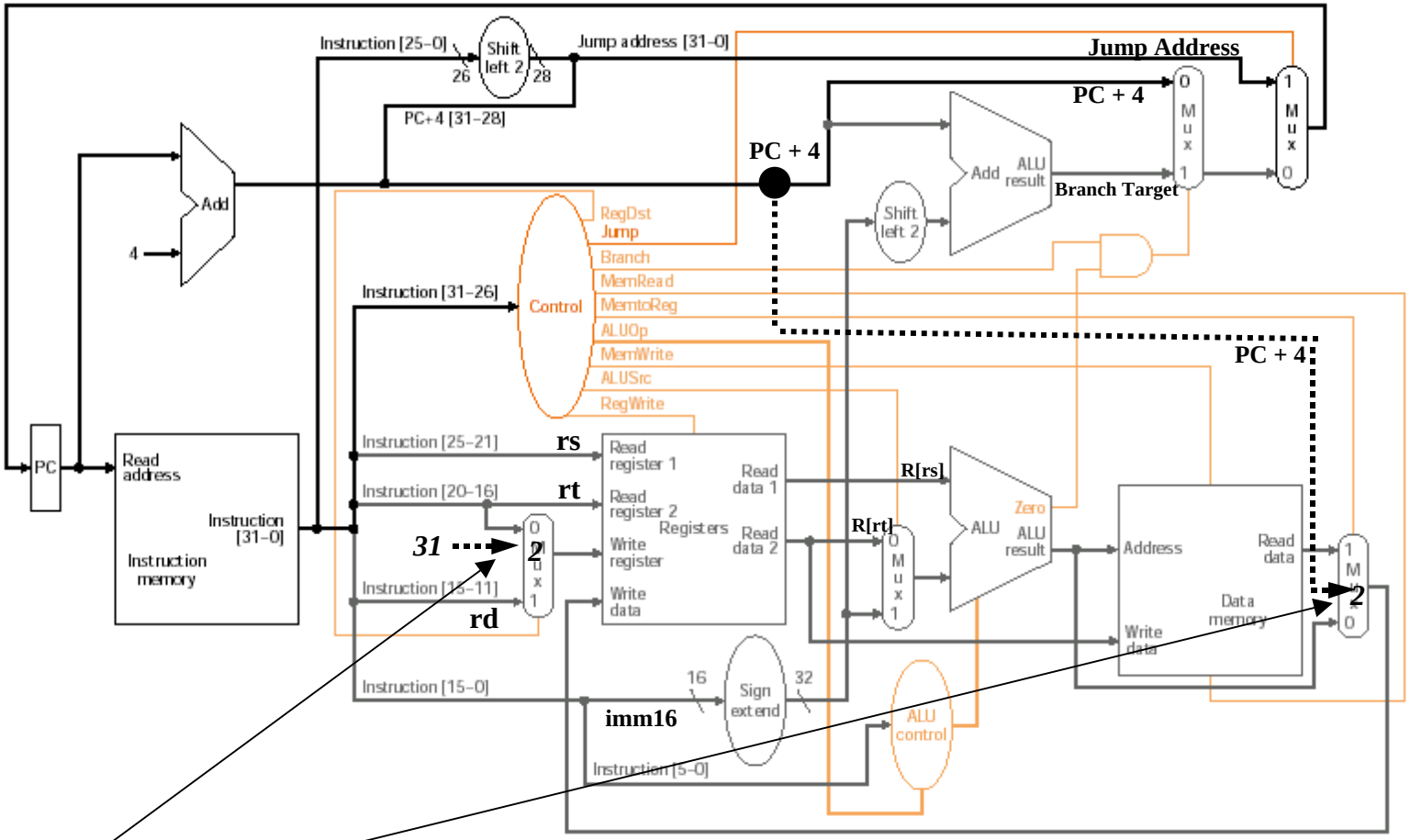
  jal Address

- **jal uses the j instruction format:**

- **We wish to add jal to the single cycle datapath in Figure 5.24 page 314. Add any necessary datapaths and control signals to the single-clock datapath and justify the need for the modifications, if any.**
- **Specify control line values for this instruction.**

Exercise 5.20: jump and link, jal support to Single Cycle Datapath

Instruction Word ← Mem[PC]
R[31] ← PC + 4
PC ← Jump Address



1. **Expand the multiplexor controlled by RegDst to include the value 31 as a new input 2.**
2. **Expand the multiplexor controlled by MemtoReg to have PC+4 as new input 2.**

EECC550 - Shaaban

(For More Practice Exercise 5.20)

Exercise 5.20: jump and link, jal support to Single Cycle Datapath

Adding Control Lines Settings for jal

(For Textbook Single Cycle Datapath including Jump)

RegDst Is now 2 bits

MemtoReg Is now 2 bits

	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	Jump
R-format	01	0	00	1	0	0	0	1	0	0
lw	00	1	01	1	1	0	0	0	0	0
sw	xx	1	xx	0	0	1	0	0	0	0
beq	xx	0	xx	0	0	0	1	0	1	0
J	xx	x	xx	0	0	0	x	x	x	1
JAL	10	x	10	1	0	0	x	x	x	1

R[31]

PC+ 4

PC ← Jump Address

Instruction Word ← Mem[PC]
R[31] ← PC + 4
PC ← Jump Address

(For More Practice Exercise 5.20)

Adding Support for LWR to Single Cycle Datapath (For More Practice Exercise 5.22)

- **We wish to add a variant of lw (load word) let's call it LWR to the single cycle datapath in Figure 5.24 page 314.**

LWR $rd, $rs, $rt

- **The LWR instruction is similar to lw but it sums two registers (specified by $rs, $rt) to obtain the effective load address and uses the R-Type format**
- **Add any necessary datapaths and control signals to the single cycle datapath and justify the need for the modifications, if any.**
- **Specify control line values for this instruction.**

Exercise 5.22: LWR (R-format LW) support to Single Cycle Datapath

Instruction Word ← Mem[PC]

PC ← PC + 4

R[rd] ← Mem[R[rs] + R[rt]]

No new components or connections are needed for the datapath just the proper control line settings

Adding Control Lines Settings for LWR

(For Textbook Single Cycle Datapath including Jump)

	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	Jump	
R-format	1	0	0	1	0	0	0	1	0	0	
lw	0	1	1	1	1	0	0	0	0	0	
sw	x	1	x	0	0	1	0	0	0	0	
beq	x	0	x	0	0	0	1	0	1	0	
J	x	x	x	0	0	0	x	x	x	1	
LWR	1	0	1	1	1	0	0	0	0	0	

rd R[rt]

Add

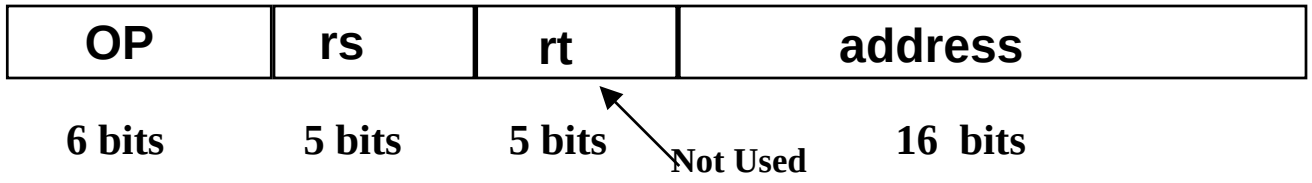
Adding Support for jm to Single Cycle Datapath

(Based on "For More Practice Exercise 5.44" but for single cycle)

- **We wish to add a new instruction jm (jump memory) to the single cycle datapath in Figure 5.24 page 314.**

  jm offset($rs)

- **The jm instruction loads a word from effective address ($rs + offset), this is similar to lw except the loaded word is put in the PC instead of register $rt.**
- **Jm used the I-format with field rt not used.**



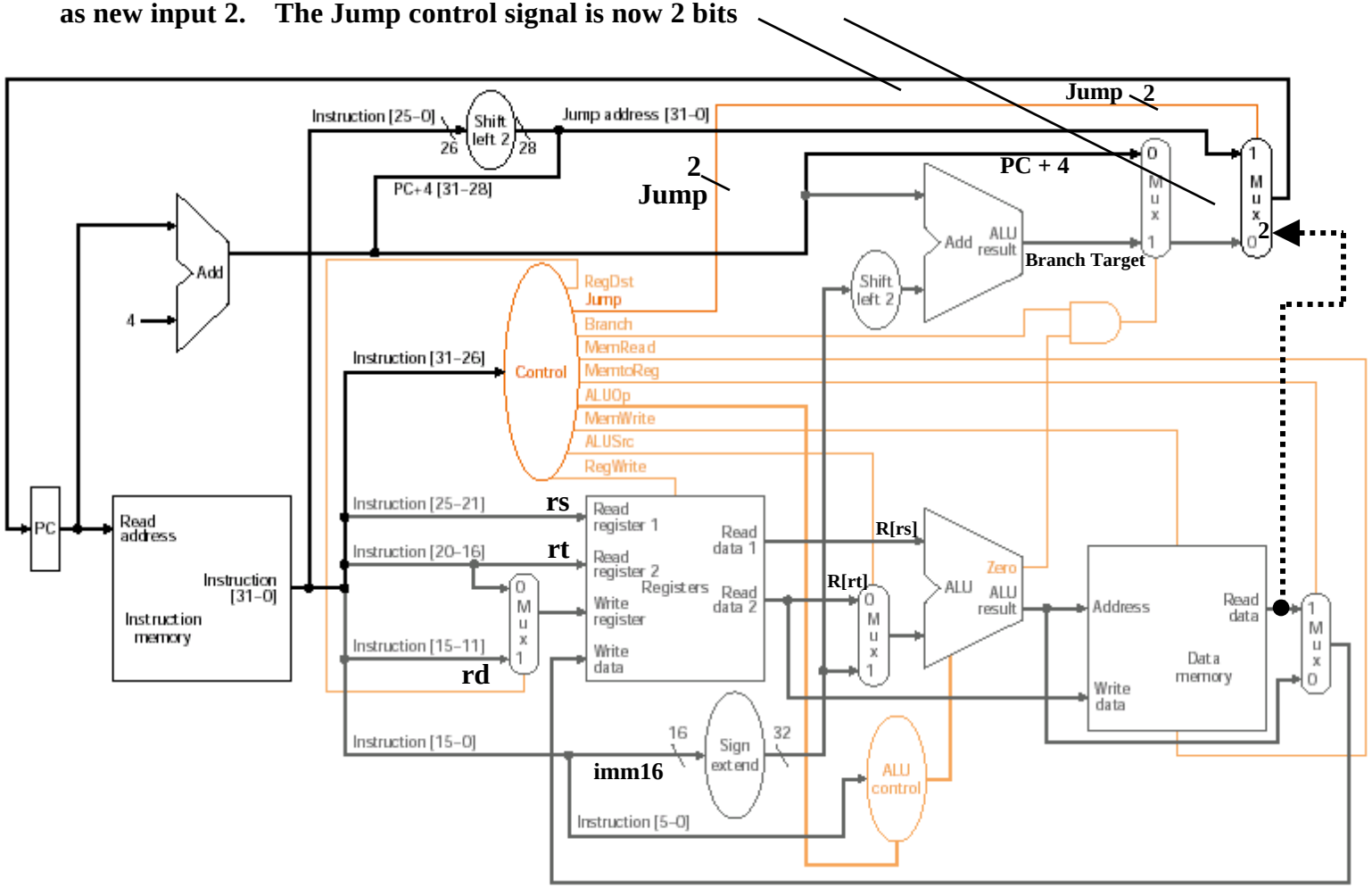
- **Add any necessary datapaths and control signals to the single cycle datapath and justify the need for the modifications, if any.**
- **Specify control line values for this instruction.**

Adding jump memory, jm support to Single Cycle Datapath

Instruction Word ← Mem[PC]

PC ← Mem[R[rs] + SignExt[imm16]]

1. **Expand the multiplexor controlled by Jump to include the Read Data (data memory output) as new input 2. The Jump control signal is now 2 bits**



(Based on "For More Practice Exercise 5.44" but for single cycle)

Adding Control Lines Settings for jm

(For Textbook Single Cycle Datapath including Jump)

Jump
is now 2 bits

	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	Jump	
R-format	1	0	0	1	0	0	0	1	0	00	
lw	0	1	1	1	1	0	0	0	0	00	
sw	x	1	x	0	0	1	0	0	0	00	
beq	x	0	x	0	0	0	1	0	1	00	
J	x	x	x	0	0	0	x	x	x	01	
JAL	x	1	x	0	1	0	x	0	0	10	

R[rs]

add

PC ← Mem[R[rs] + SignExt[imm16]]

Adding Support for swap to Multi Cycle Datapath

(For More Practice Exercise 5.42)

- **You are to add support for a new instruction, swap that exchanges the values of two registers to the MIPS multicycle datapath of Figure 5.28 on page 232**

  swap $rs, $rt

- **Swap used the R-Type format with:**

  the value of field rs = the value of field rd

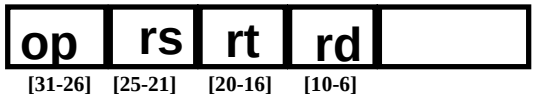
- **Add any necessary datapaths and control signals to the multicycle datapath. Find a solution that minimizes the number of clock cycles required for the new instruction without modifying the register file. Justify the need for the modifications, if any.**
- **Show the necessary modifications to the multicycle control finite state machine of Figure 5.38 on page 339 when adding the swap instruction. For each new state added, provide the dependent RTN and active control signal values.**

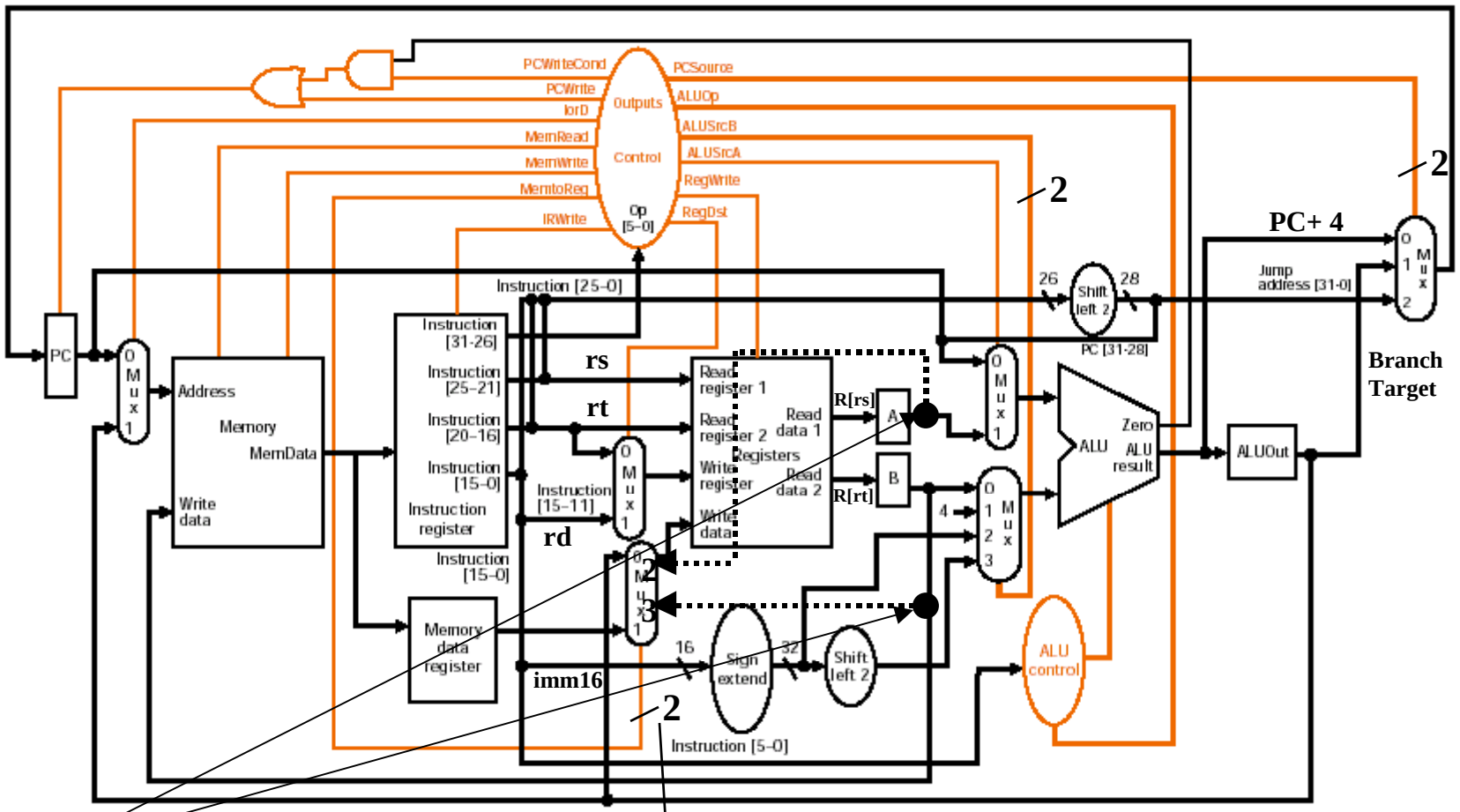
Adding swap Instruction Support to Multi Cycle Datapath

Swap $rs, $rt **R[rt] ← R[rs]**

R[rs] ← R[rt]

We assume here rs = rd in instruction encoding

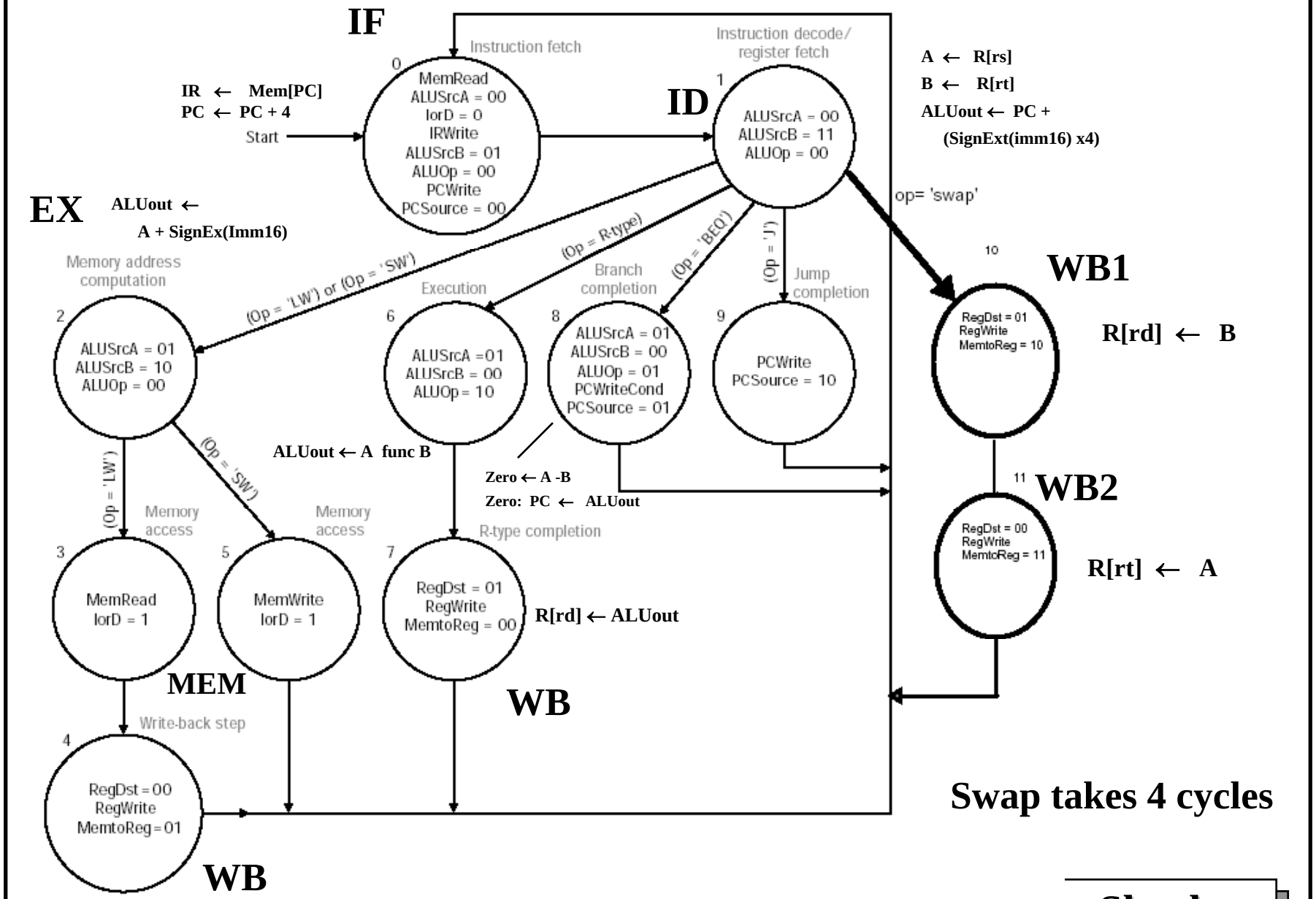




The outputs of A and B should be connected to the multiplexor controlled by MemtoReg if one of the two fields (rs and rd) contains the name of one of the registers being swapped. The other register is specified by rt. The MemtoReg control signal becomes two bits.

(For More Practice Exercise 5.42)

Adding swap Instruction Support to Multi Cycle Datapath



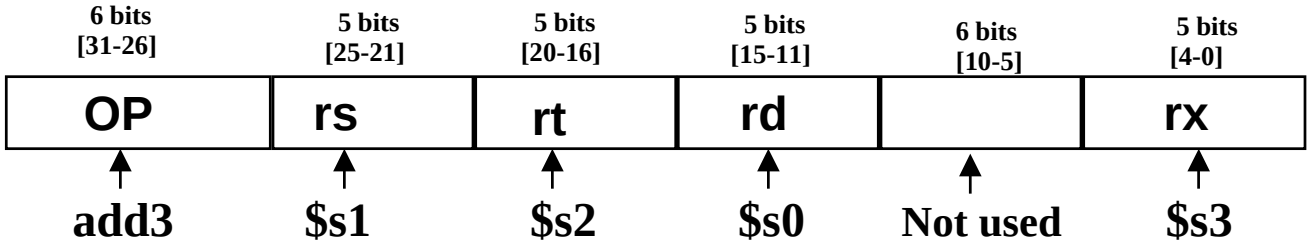
Adding Support for add3 to Single Cycle Datapath (For More Practice Exercise 5.45)

- **You are to add support for a new instruction, add3, that adds the values of three registers, to the MIPS multicycle datapath of Figure 5.28 on page 232**
  For example:

  add3 $s0,$s1, $s2, $s3

  Register $s0 gets the sum of $s1, $s2 and $s3.

  The instruction encoding uses a modified R-format, with an additional register specifier rx added replacing the five low bits of the "funct" field.



- **Add necessary datapath components, connections, and control signals to the multicycle datapath <u>without modifying the register bank or adding additional ALUs</u>. Find a solution that minimizes the number of clock cycles required for the new instruction. Justify the need for the modifications, if any.**
- **Show the necessary modifications to the multicycle control finite state machine of Figure 5.38 on page 339 when adding the add3 instruction. For each new state added, provide the dependent RTN and active control signal values.**

Exercise 5.45: add3 instruction support to Multi Cycle Datapath

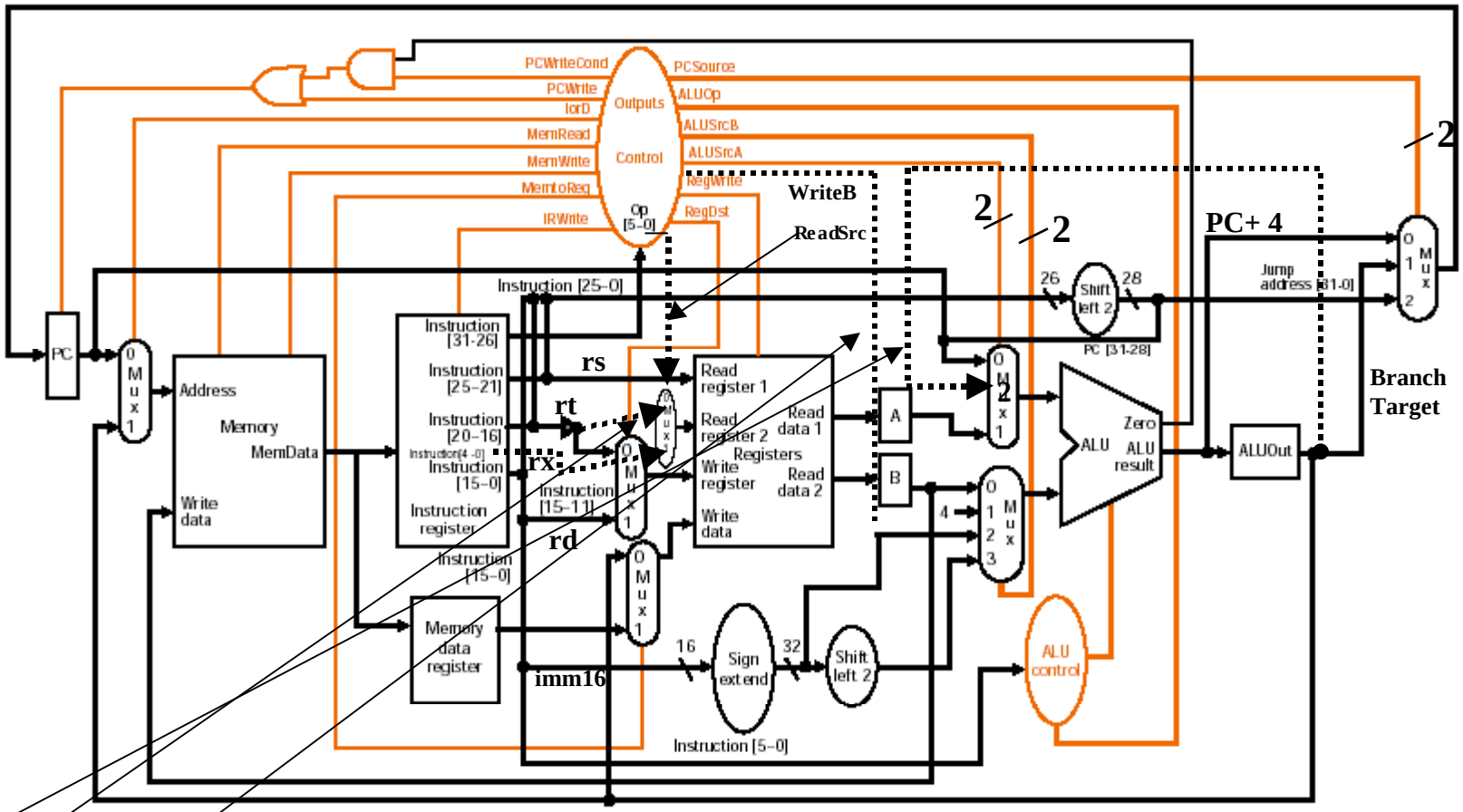
Add3 $rd, $rs, $rt, $rx

R[rd] ← R[rs] + R[rt] + R[rx]

rx is a new register specifier in field [0-4] of the instruction
No additional register read ports or ALUs allowed

Modified R-Format

op	rs	rt	rd		rx
[31-26]	[25-21]	[20-16]	[10-6]		[4-0]



1. ALUout is added as an extra input to first ALU operand MUX to use the previous ALU result as an input for the second addition.
2. A multiplexor should be added to select between rt and the new field rx containing register number of the 3rd operand (bits 4-0 for the instruction) for input for Read Register 2.
This multiplexor will be controlled by a new one bit control signal called ReadSrc.
3. WriteB control line added to enable writing R[rx] to B

Exercise 5.45: add3 instruction support to Multi Cycle Datapath