IT5002 Tutorial 3

AY 2025/26 Semester 1

Prepared by Michael Yang

Slides adapted from Theodore Leebrant, Prof. Colin and Prof. Aaron

Q1. Datapath and Control

For each of the following instructions, figure out what data is being used at various points in the datapath and what control signals are generated

i. 0x8df80000: lw $24, 0($15)

ii. 0x1023000C: beq $1, $3, 12

iii. 0x0285c822: sub $25, $20, $5

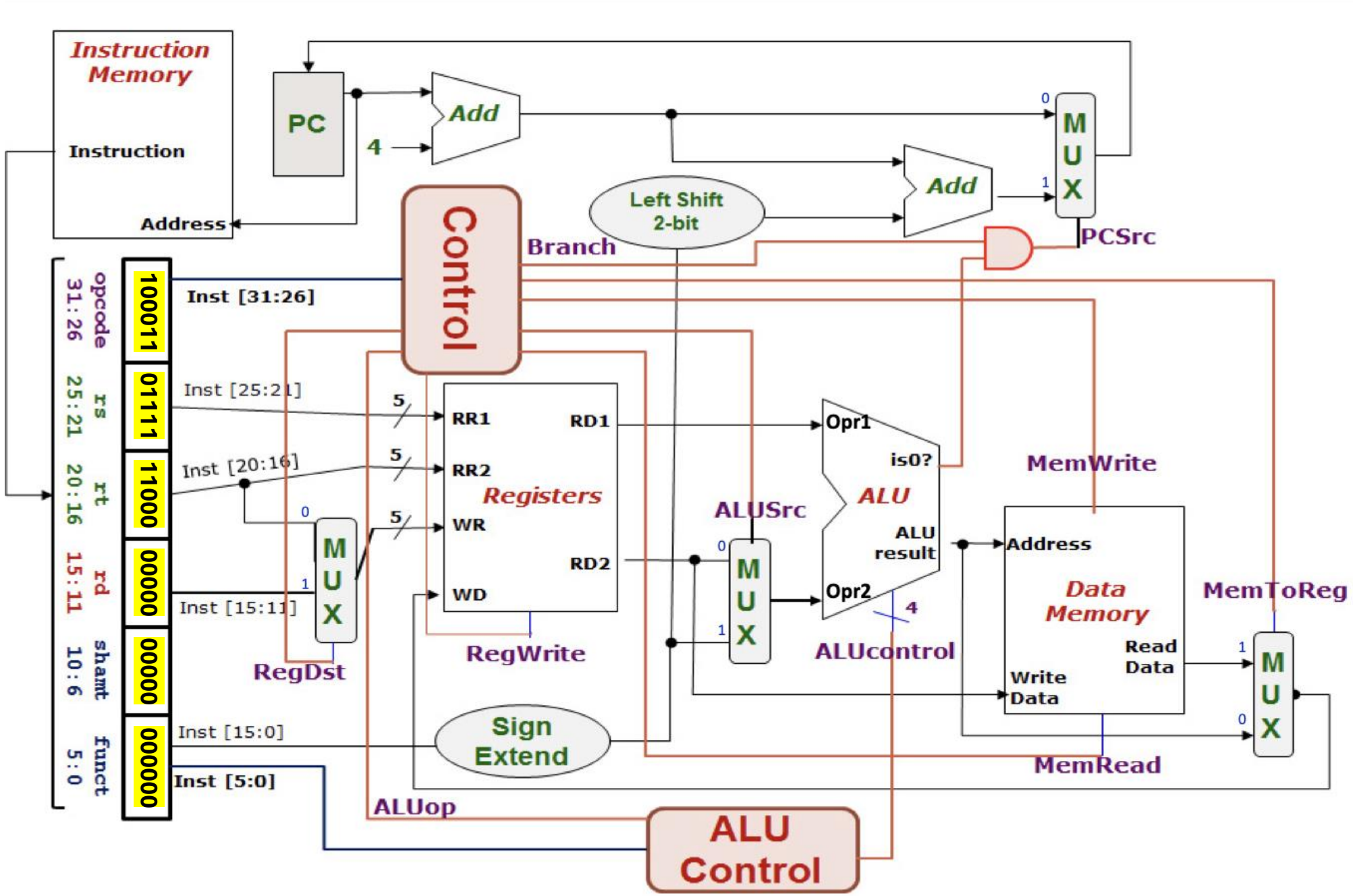
Outputs of **Control Unit** for each type of instruction

	RegDst	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUop	
								op1	op2
R-type	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Outputs of **ALU Control**

	ALUop		**Funct Field (F[5:0] == Inst[5:0])**						**ALU control**
	MSB	LSB	F5	F4	F3	F2	F10	F0	
lw	0	0	X	X	X	X	X	X	0010
sw	0	0	X	X	X	X	X	X	0010
beq	X	1	X	X	X	X	X	X	0110
add	1	X	X	X	0	0	0	0	0010
sub	1	X	X	X	0	0	1	0	0110
and	1	X	X	X	0	1	0	0	0000
or	1	X	X	X	0	1	0	1	0001
slt	1	X	X	X	1	0	1	0	0111

0x8df80000: lw $24, 0($15) = 1000 1101 1111 1000 0000 0000 0000 0000



0x8df80000: lw $24, 0($15)

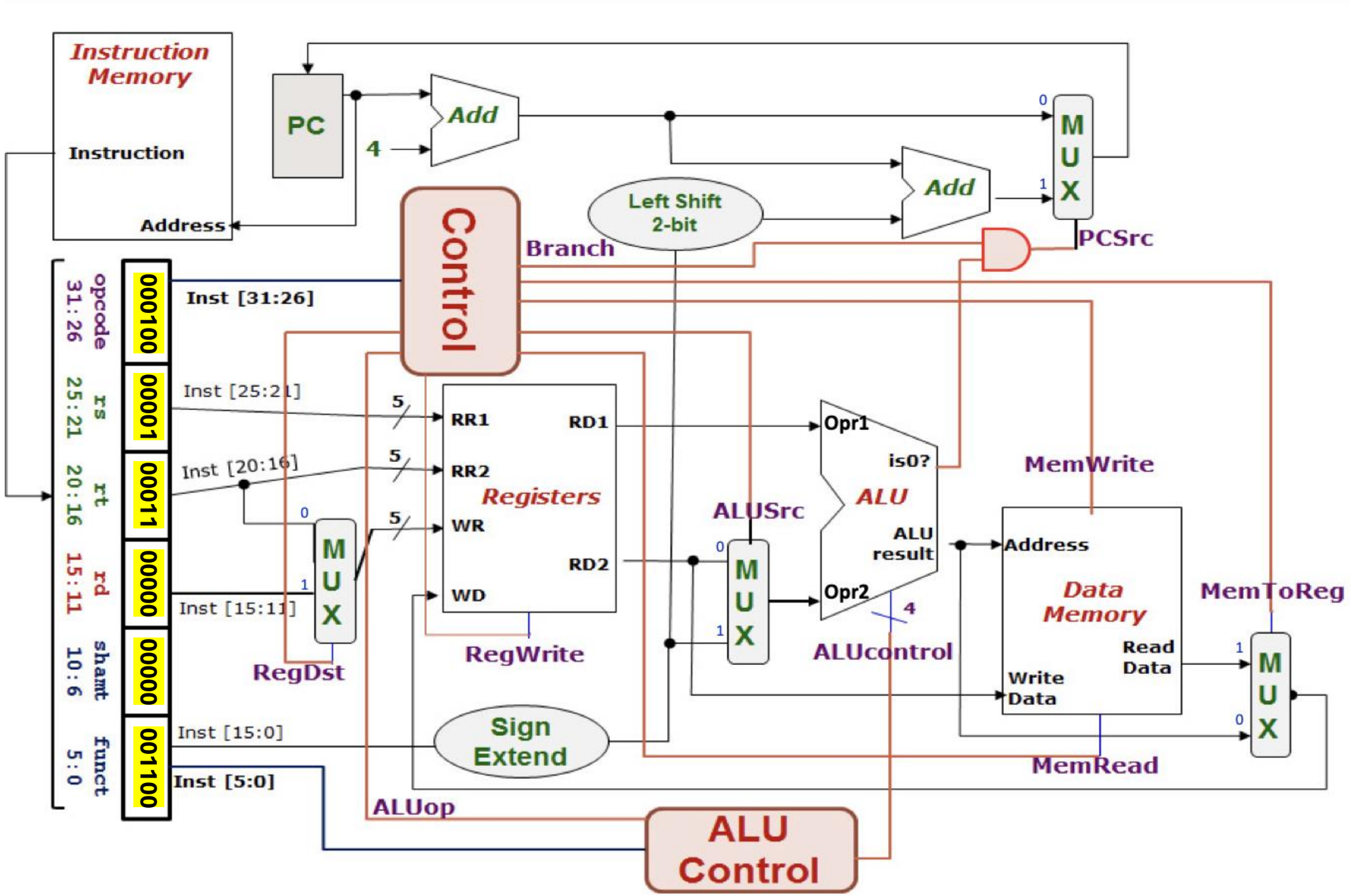
($x means register, [$x] means value stored in register,
Mem(...) means memory stored at that location)

RegDst	RegWrite	ALUSrc	MemRead	MemWrite	MemToReg	Branch	ALUop	ALUcontrol
0	1	1	1	0	1	0	00	0010

Register Files				ALU		Data Memory	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data
$15	$24	$24	Mem([$15]+0)	[$15]	0	[$15]+0	[$24]

Next PC = PC+4

0x1023000C = beq $1, $3, 12 = 0001 0000 0010 0011 0000 0000 0000 1100



0x1023000C = beq $1, $3, 12

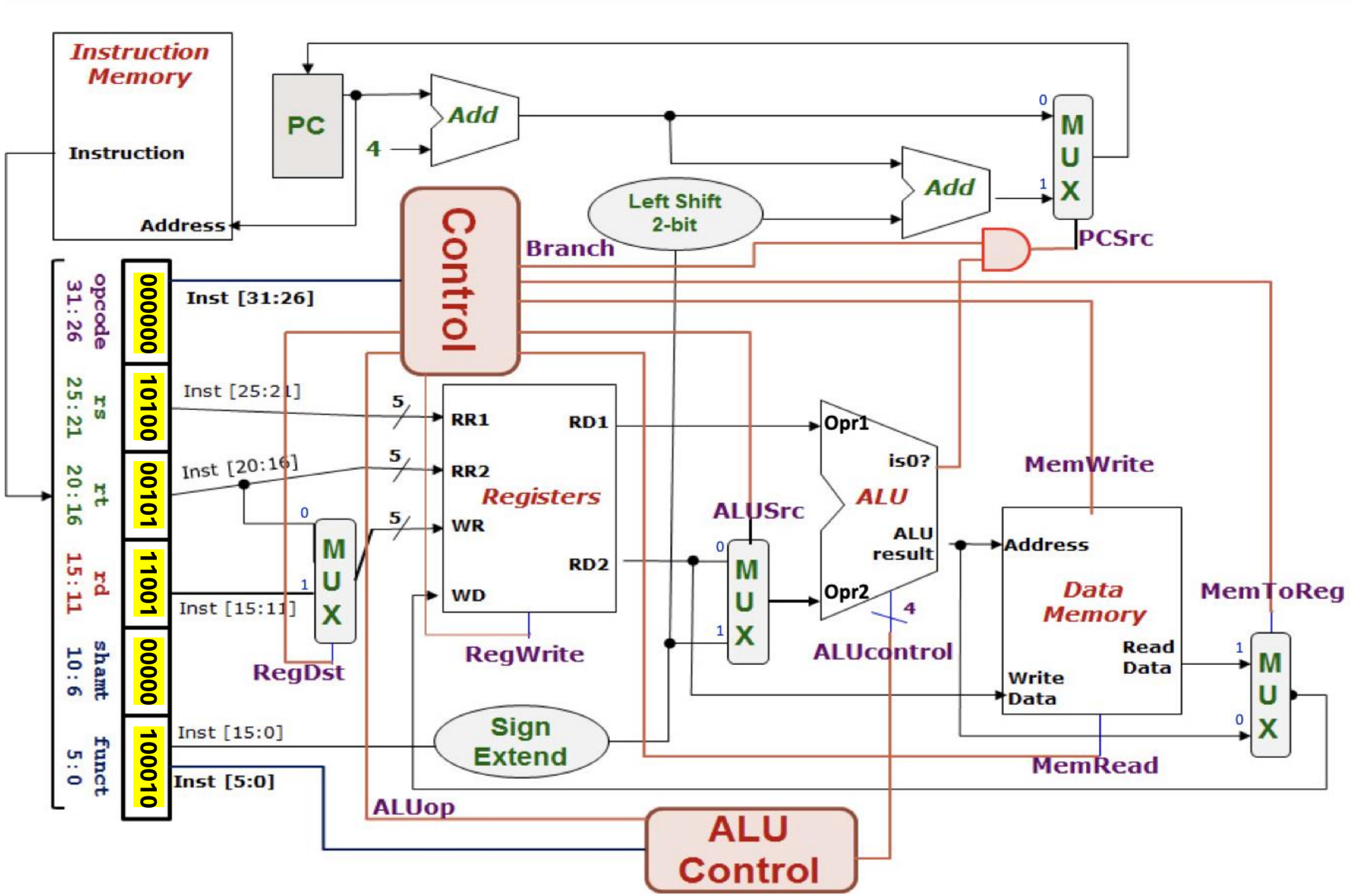
($x means register, [$x] means value stored in register,
Mem(…) means memory stored at that location)

RegDst	RegWrite	ALUSrc	MemRead	MemWrite	MemToReg	Branch	ALUop	ALUcontrol
X	0	0	0	0	X	1	01	0110

Register Files				ALU		Data Memory	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data
$1	$3	$3 or $0	[$1]-[$3] or random value	[$1]	[$3]	[$1]-[$3]	[$3]

Next PC = PC+4 or (PC+4)+(12x4)

0x0285c822 = sub $25, $20, $5 = 0000 0010 1000 0101 1100 1000 0010 0010



0x0285c822 = sub $25, $20, $5

($x means register, [$x] means value stored in register,
Mem(...) means memory stored at that location)

RegDst	RegWrite	ALUSrc	MemRead	MemWrite	MemToReg	Branch	ALUop	ALUcontrol
1	1	0	0	0	0	0	10	0110

Register Files				ALU		Data Memory	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data
$20	$5	$25	[$20]-[$5]	[$20]	[$5]	[$20]-[$5]	[$5]

Next PC = PC+4

Q2. Latency and Critical Path

Given the latencies of each hardware component,

Inst-Mem	Adder	MUX	ALU	Reg-File	Data-Mem	Control/ALU control	Left-shift / Sign extend / AND
400 ps	100 ps	30 ps	120 ps	200 ps	350 ps	100 ps	20 ps

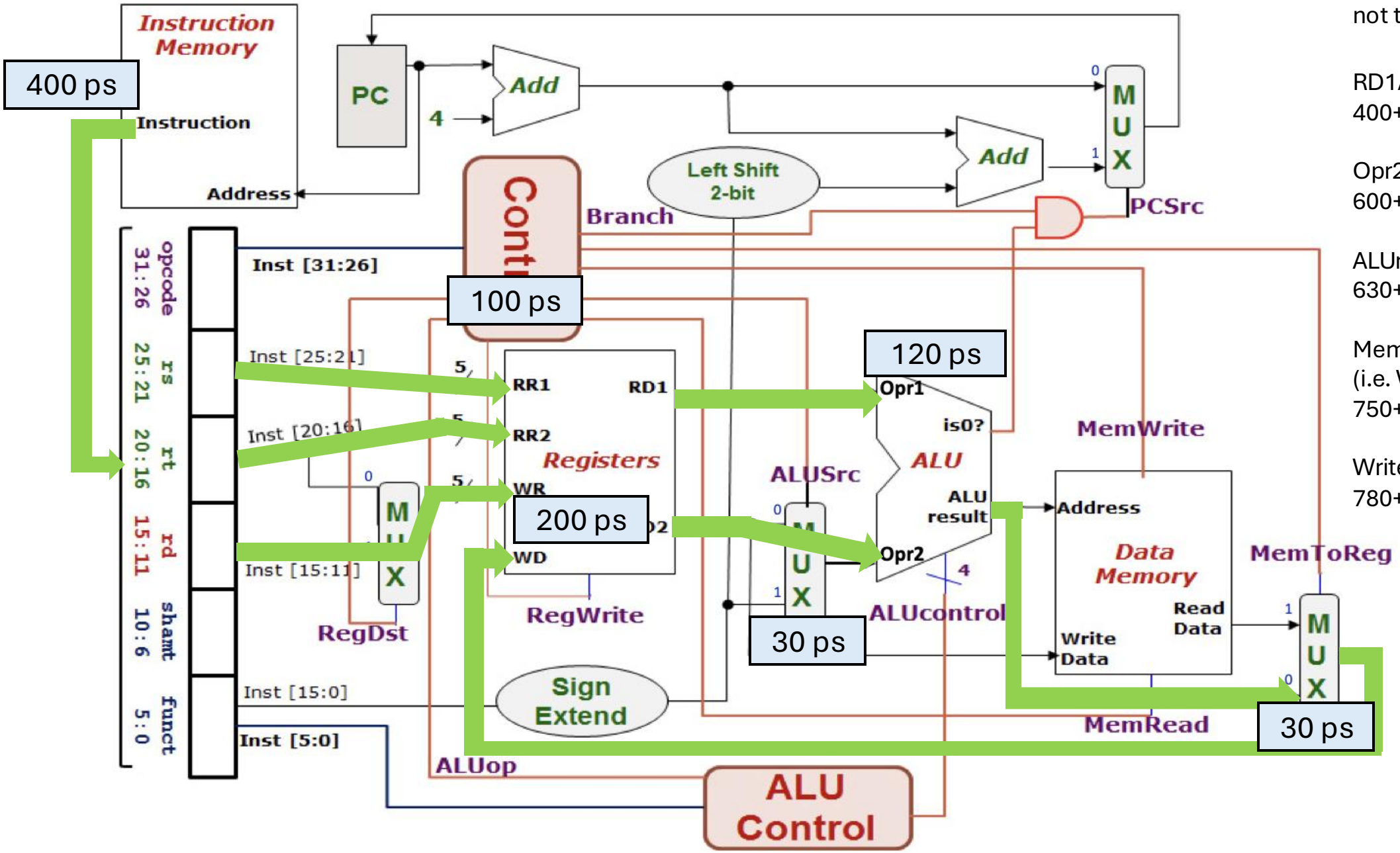
Estimate the latency for each of the following instructions:

i. sub (e.g. sub $25, $20, $5)

ii. lw (e.g. lw $24, 0($15))

iii. beq (e.g. beq $1, $3, 12)

SUB instruction



Control signals are ready by 400+100=500ps (i.e. not the critical path)

RD1/RD2/Opr1 ready by 400+200=600ps

Opr2 ready by 600+30=630ps

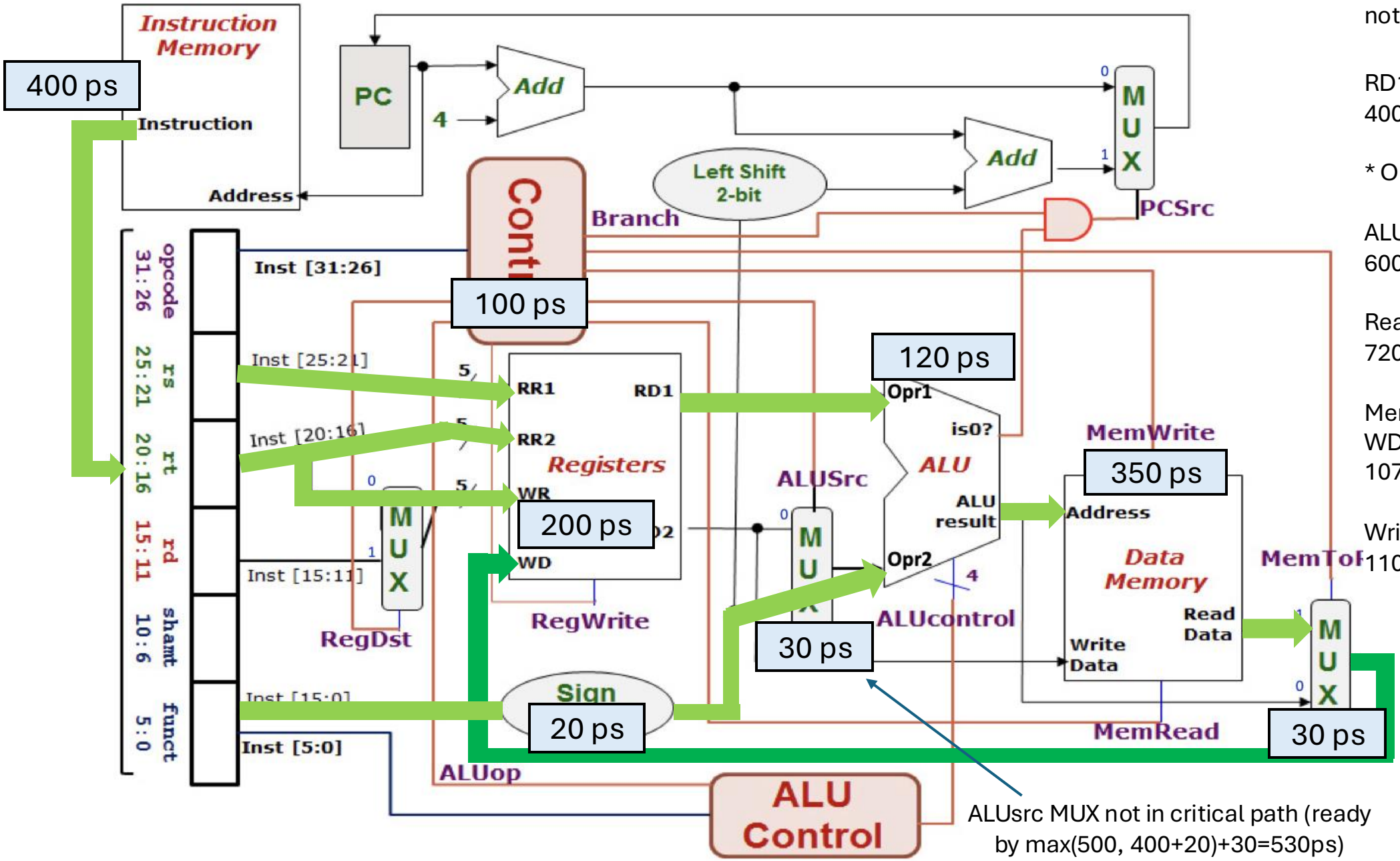
ALUresult ready by 630+120=750ps

MemToReg MUX output (i.e. Write Data) ready by 750+30=780ps

Write to WR done by 780+200=980ps

400+200+30+ 120+30+200 = 980ps

LW instruction



Control signals are ready by 400+100=500ps (i.e. not the critical path)

RD1/RD2/Opr1 ready by 400+200=600ps

* Opr2 ready by 530ps

ALUresult ready by 600+120=720ps

Read Data ready by 720+350=1070ps

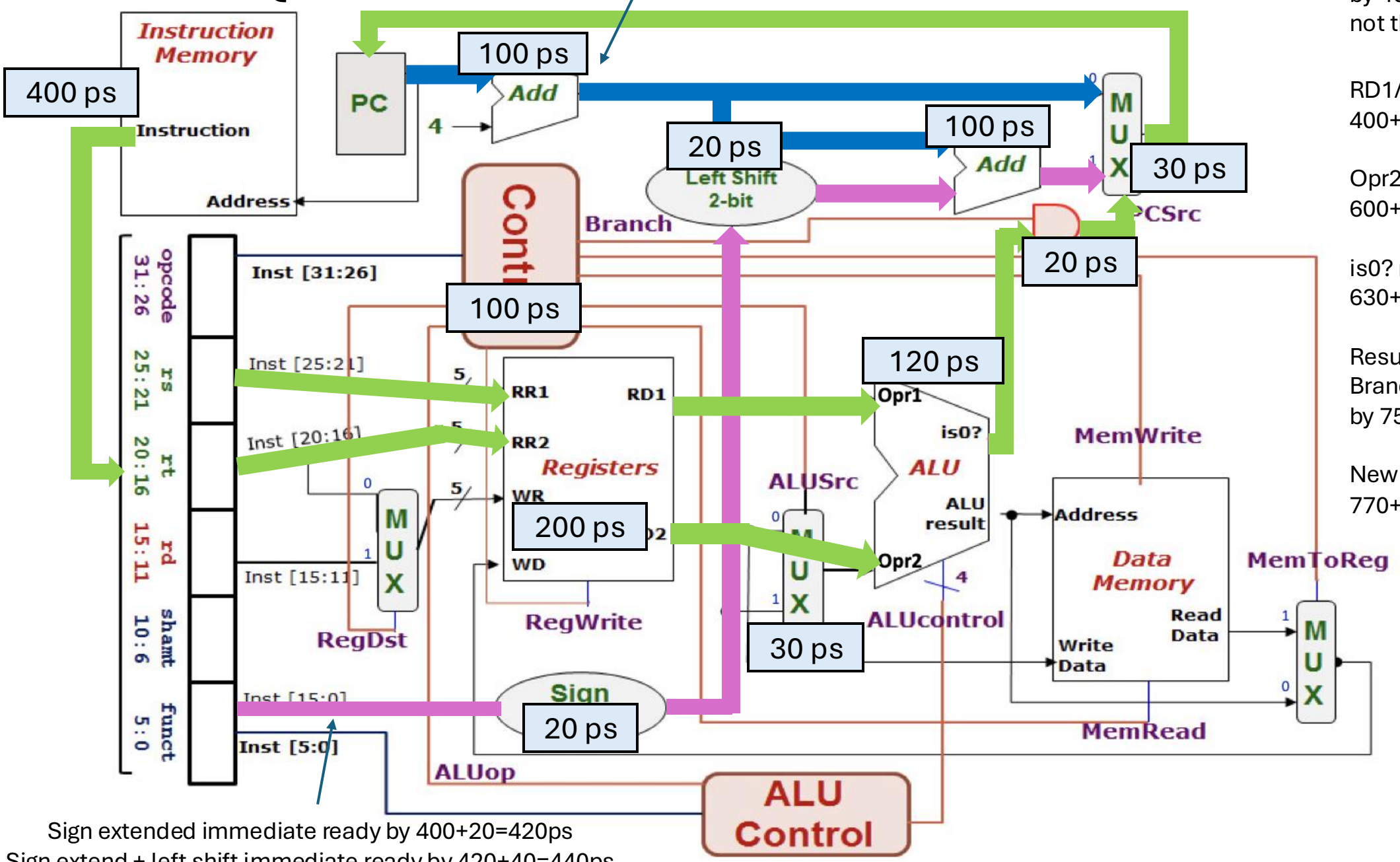
MemToReg output (i.e. WD) ready by 1070+30=1100ps

Write to register done by 1100+200=1300ps

ALUsrc MUX not in critical path (ready by max(500, 400+20)+30=530ps)

400+200+120 +350+30+200 = 1300ps

BEQ instruction



Control signals are ready by 400+100=500ps (i.e. not the critical path)

RD1/RD2/Opr1 ready by 400+200=600ps

Opr2 ready by 600+30=630ps

is0? ready by 630+120=750ps

Result of (is0? AND Branch) i.e. PCSrc ready by 750+20=770ps

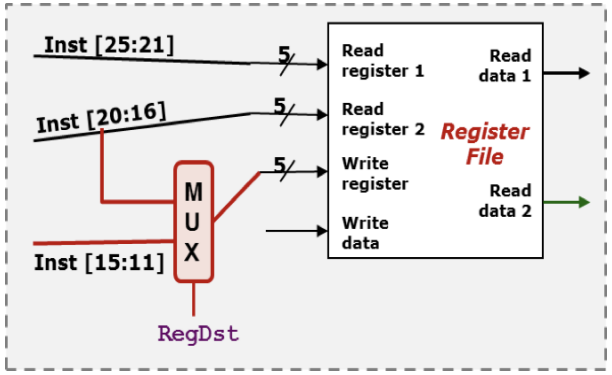
New PC value ready by 770+30=800ps

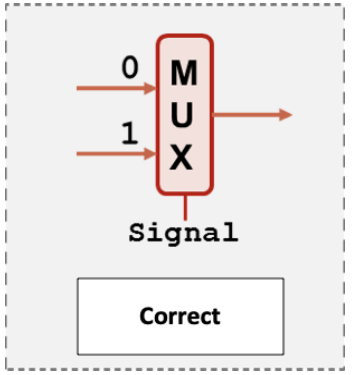
400+200+30+ 120+20+30 =800ps

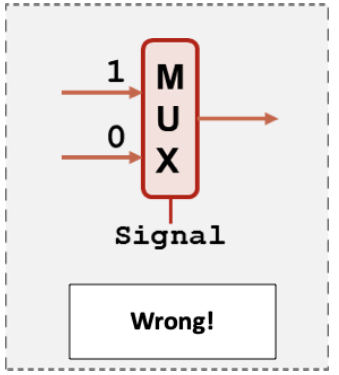
Sign extended immediate ready by 400+20=420ps

Sign extend + left shift immediate ready by 420+40=440ps

Q3. Swapped Inputs







The inputs to the RegDst multiplexor were swapped.

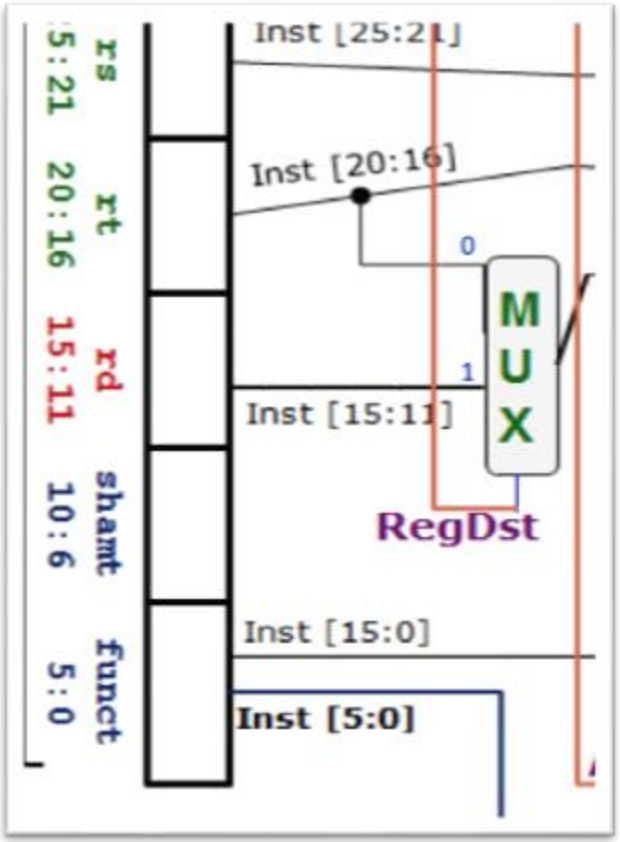
For each of the following types of instruction, give an example of
(i) an instruction that will still produce the same result as expected,
(ii) an instruction that produces a wrong result,

or state that none exist.

(a) add
(b) lw
(c) beq (provide the branch offset as the immediate value)

General strategy:

- To produce the same result, we can have
  - $rt = $rd
- So that swapping makes no difference



Add instruction

(i) `add $t0, $t1, $t0`

$rd = $t0

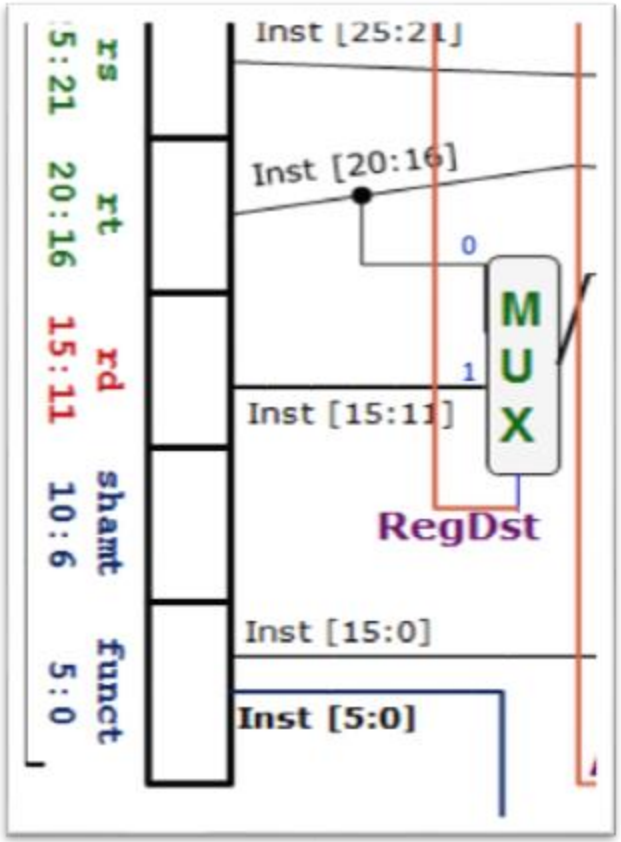
$rs = $t1

$rt = $t0

(ii) Any instruction where

$rt != $rd, e.g.

`add $t0, $t1, $t2`

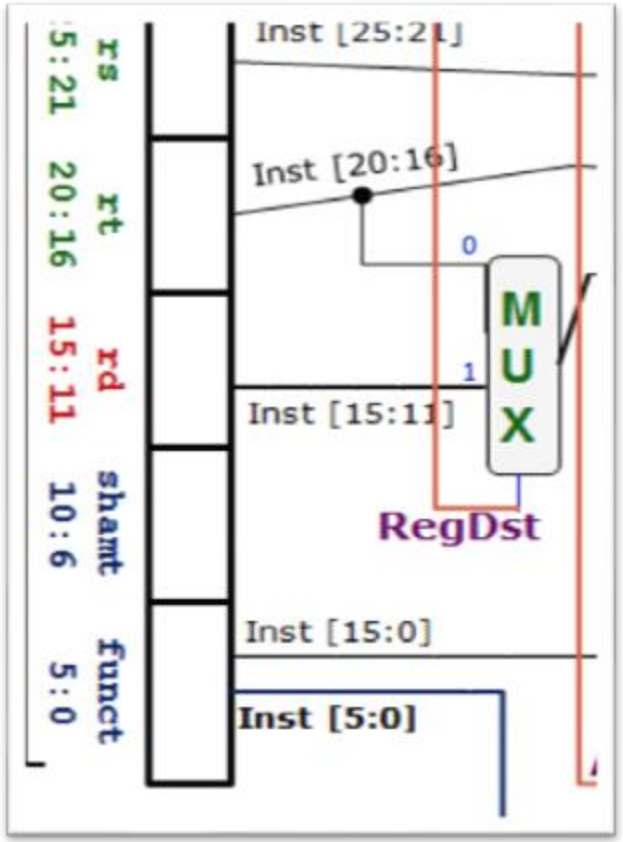


Load Word instruction

Choose the first 5 bits of imm as
00100 = 4 = $a0, and set $rt = $a0

0010 0000 0000 0000 = 2^13 = 8192

Note that $rs doesn't matter here
(i) `lw $a0, 8192($0)`



(ii) Anything where the first
5 bits of the immediate value
is not equal to the register's number
(but it still must be a valid register between 0 and 31)

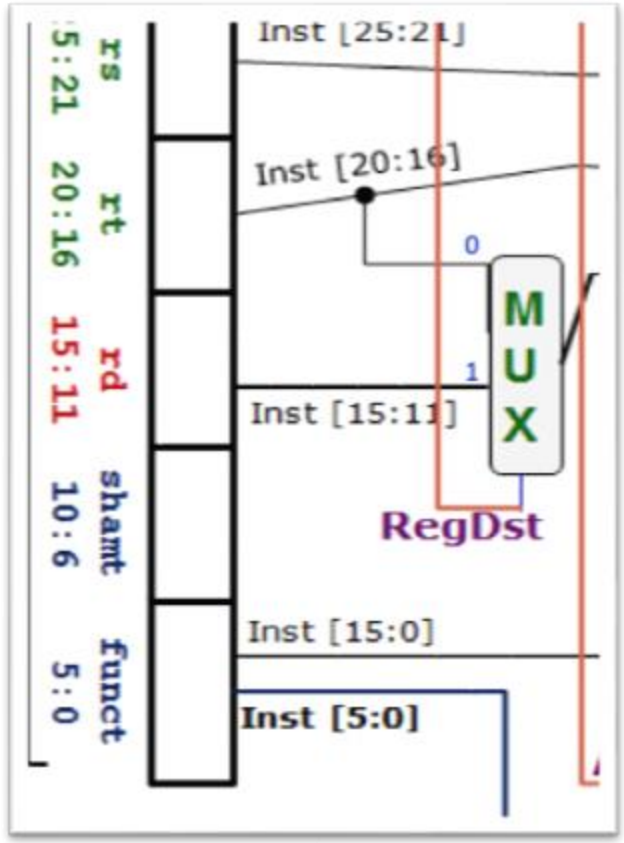
E.g. Immediate = 0000 0000 0000 0000
$rt = $t0 = 24 = 11000

Branch on equal instruction

(ii) *Any* branch instruction works

Write register (WR) is never used, so it does not matter which register RegDst selects

(ii) No answer



Slides uploaded to https://github.com/michaelyql/IT5002

Email: e1121035@u.nus.edu

Anonymous feedback: https://bit.ly/feedback-michael

(or scan the QR below)