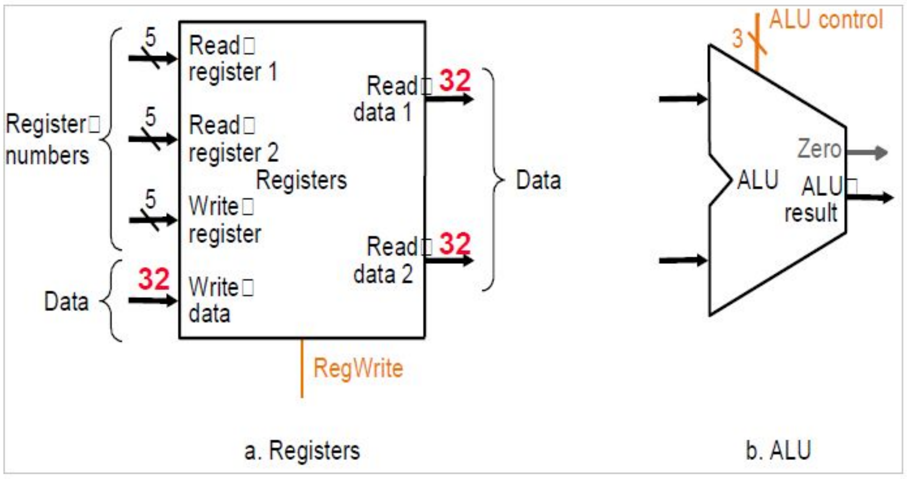
1. datapath for R-type instructions

add $r1, $r0, $r3



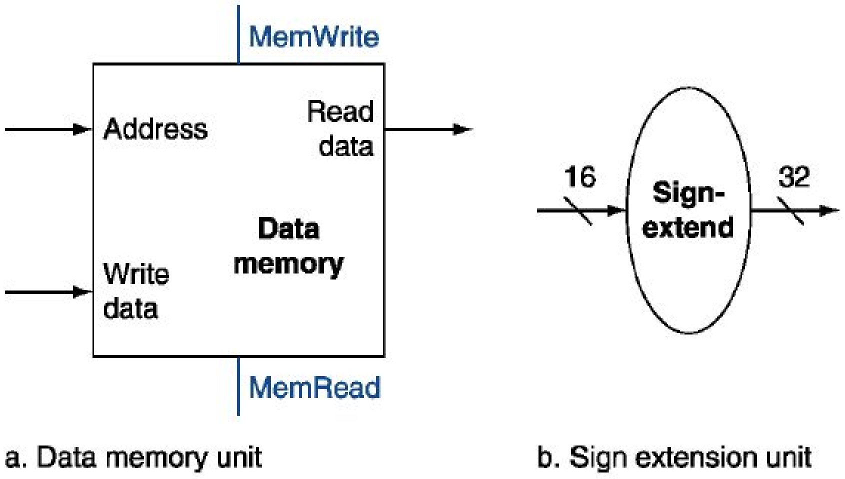
5 bits for read register numbers

32 bits for two output data and a write data

2. Load / Store instructions

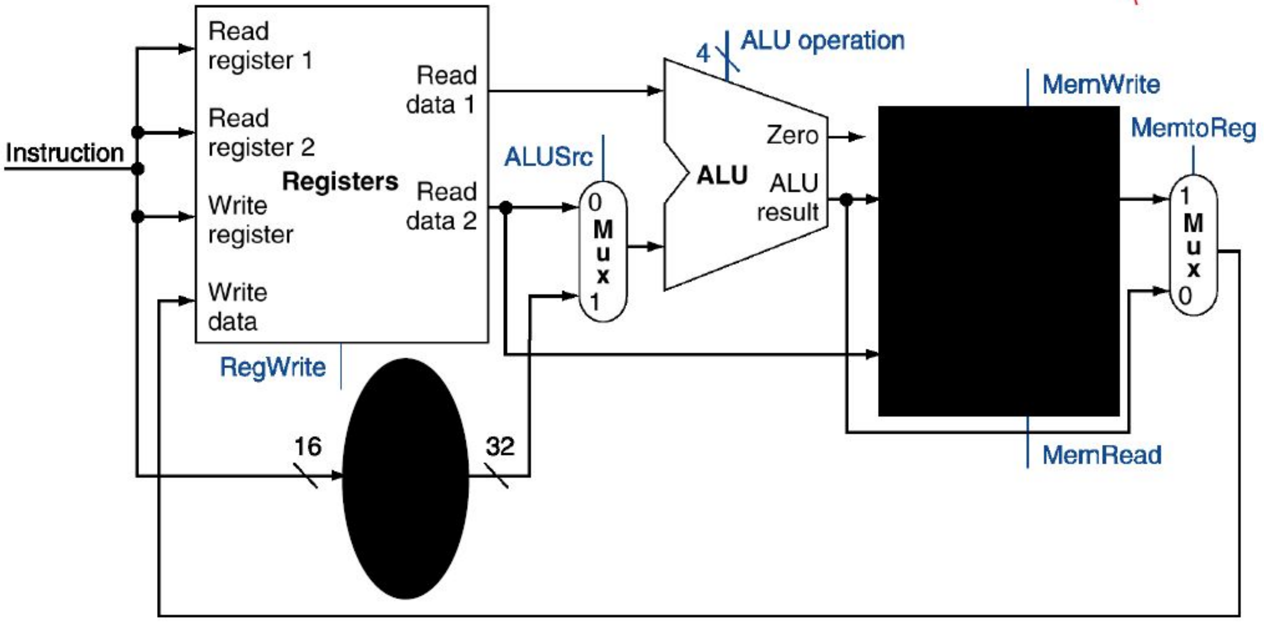
- both register files and ALU (address calculation)
- *Memory and sign extension (32-bit elements in MIPS)*

lw $s1, 100($s2) // $s1 = Memory[$s2 + 100]



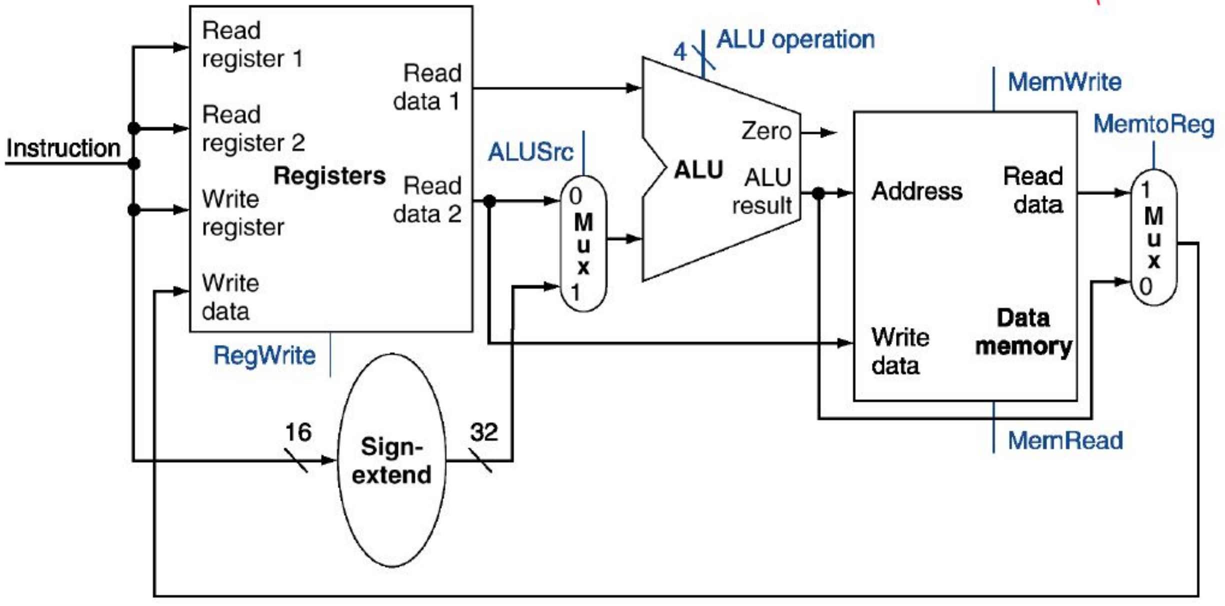
add $s1, $s2, $s3 // $s1 = $s2 + $s3

Share Target Register:
From arithmetic results
From memory

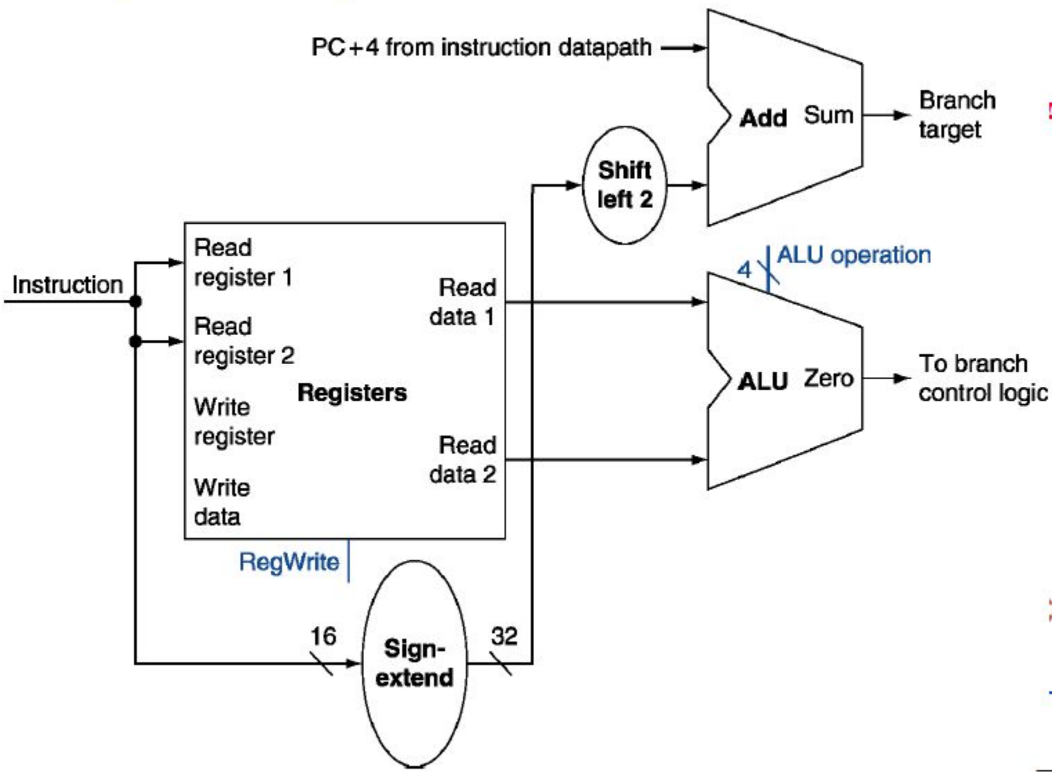


lw $s1, 100($s2) // $s1 = Memory[$s2 + 100]

Share Target Register:
From arithmetic results
From memory

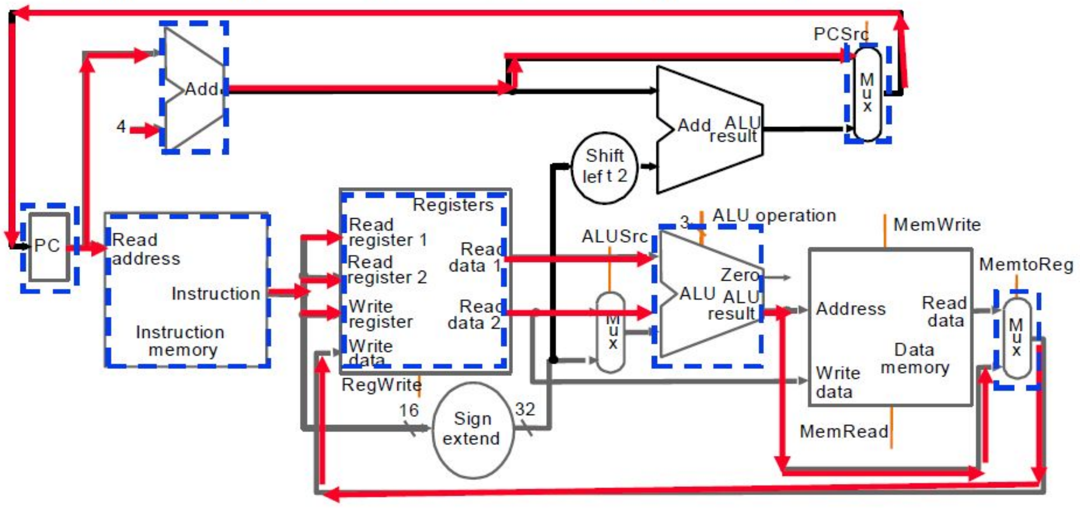


- *Two steps and operations*
  1. *Compute the branch target address*
  2. *Compare the register contents*



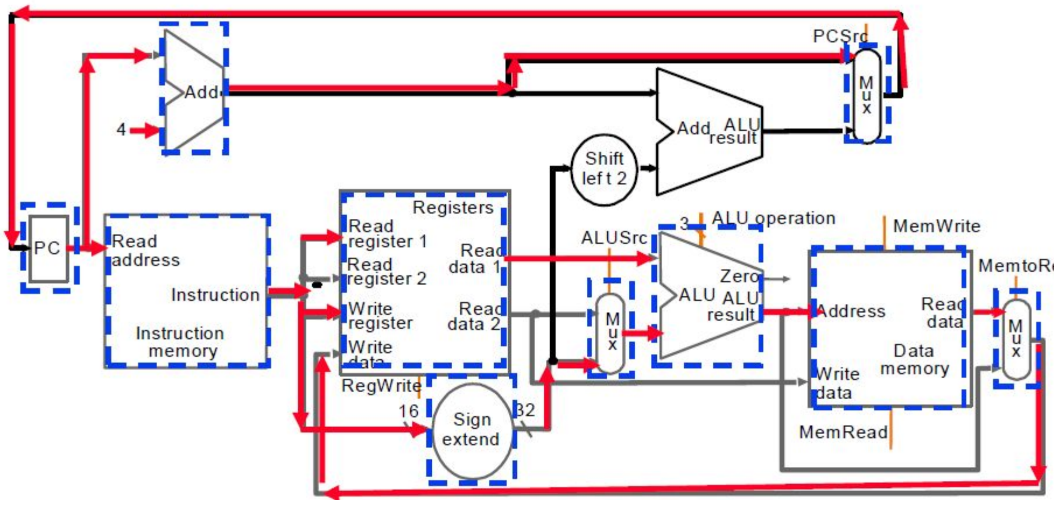
- R[rd] <- R[rs] op R[rt]   **Example**:   add rd, rs, rt
  add r3, r5, r6
  // r3 = r5 + r6

31	26	21	16	11	6	0
op	**rs**	**rt**	**rd**	**shamt**	**funct**	
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	



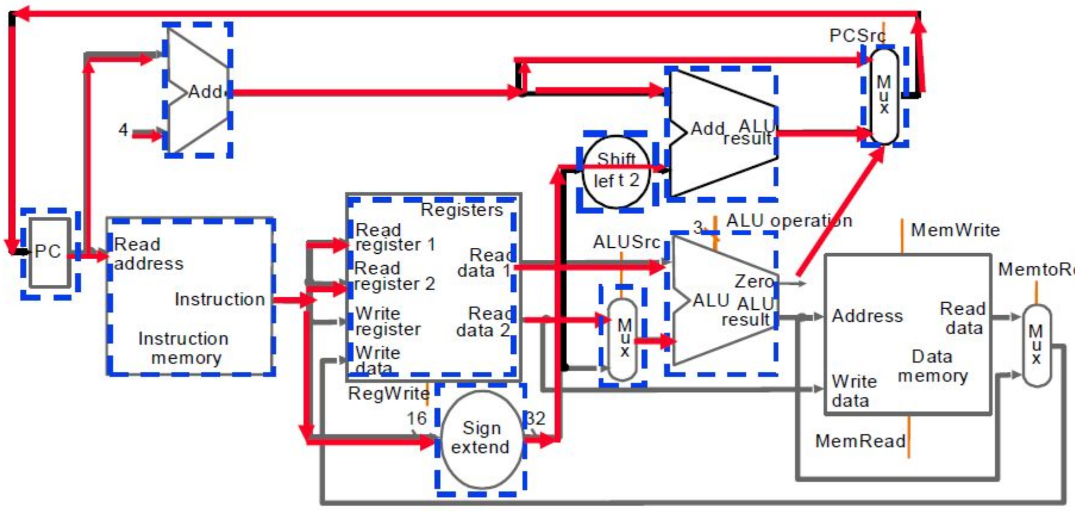
◦ R[rt] <- Mem[R[rs] + SignExt[imm16]] **Example**: lw rt, rs,
lw r7, 30(r8)

31–26	25–21	20–16	15–0
op	**rs**	**rt**	**immediate**
6 bits	5 bits	5 bits	16 bits



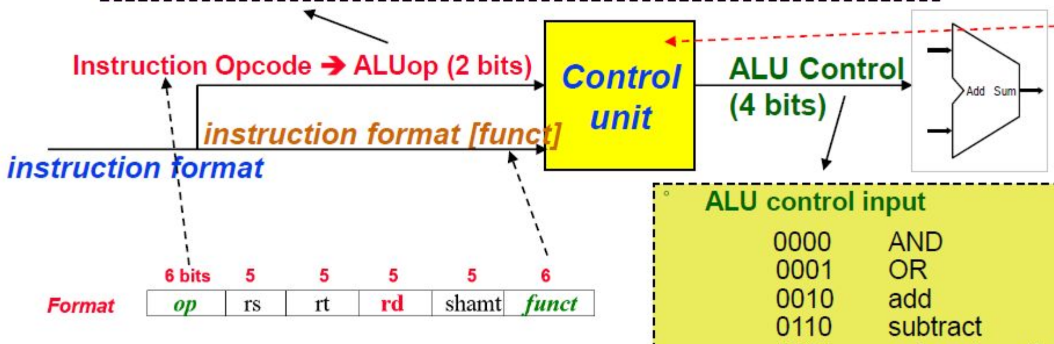
◦ beq rs, rt, imm16 beq r3,r4,1000 ***OR*** beq r3, 1000(r4)

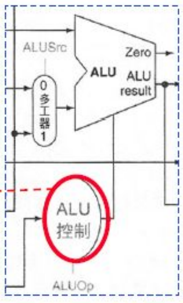
31–26	25–21	20–16	15–0
op	**rs**	**rt**	**immediate**
6 bits	5 bits	5 bits	16 bits



- *Selecting the operations to perform* (ALU, read/write, etc.)
  - Controlling the flow of data (multiplexor inputs)
  - Information comes from the 32 bits of the instruction
  - *ALU's operation based on instruction type and function code*
- Using *ALUop* to indicate instruction type

00 = lw, sw	(*add* - compute the memory address)
01 = beq	(*subtraction* - comparison)
10 = arithmetic	(*calculation*, decided from ***funct*** column)

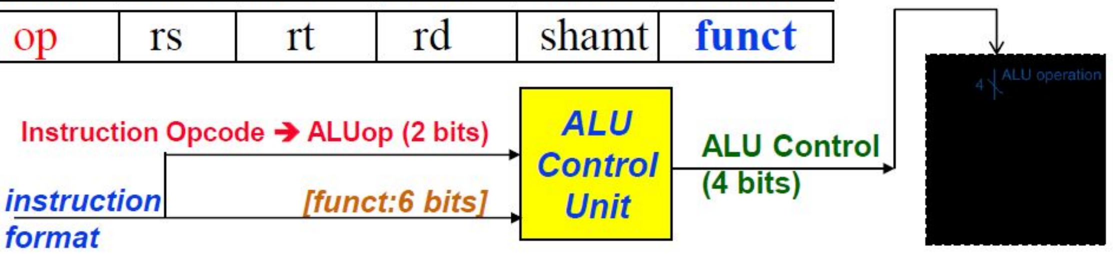




ALU control input

0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

000000	10001	10010	01001	00000	100000
op	rs	rt	rd	shamt	**funct**

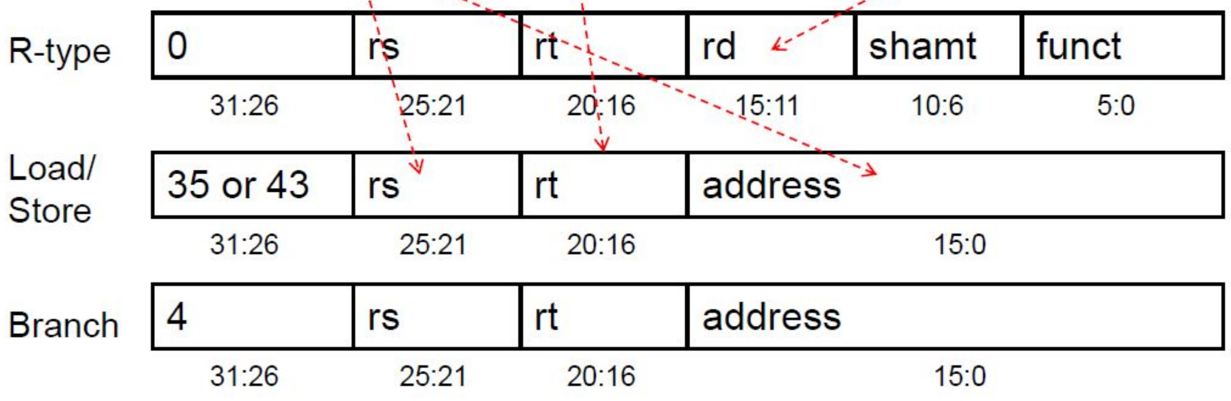


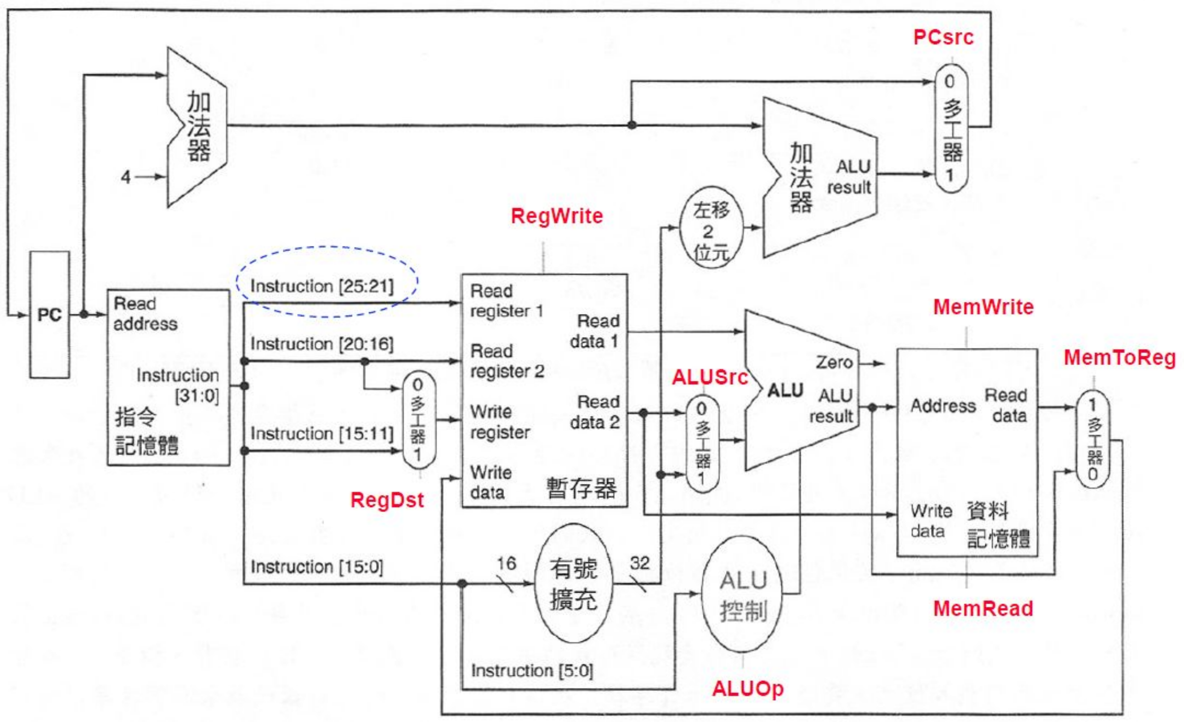
opcode	**ALUOp**	Operation	funct	ALU function	**ALU control**
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

		Input								Output
		ALUOp		Funct field						Operation
		ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
load/store word	LW/SW	0	0	X	X	X	X	X	X	010
branch	BE	X	1	X	X	X	X	X	X	110
R-type	add	1	X	X	X	0	0	0	0	010
R-type	substract	1	X	X	X	0	0	1	0	110
R-type	and	1	X	X	X	0	1	0	0	000
R-type	or	1	X	X	X	0	1	0	1	001
R-type	set on less than	1	X	X	X	1	0	1	0	111

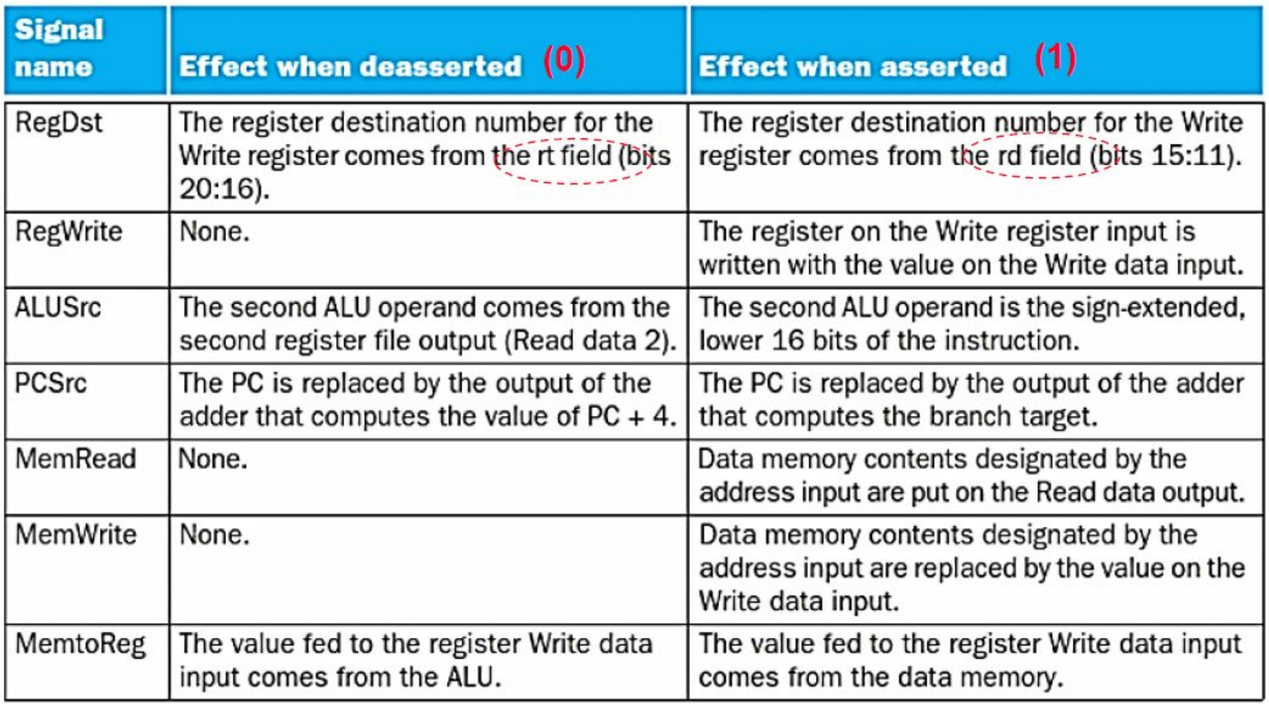
◦ Observations：

- *Op* : 31:26 (6 bits) ➔ Op[5:0]
- 2 *source registers*： 25:21(rs), 20:16(rt)
- *Base register* for load and store：25:21 (rs)
- *Offset* for branch equal, load, store：15:0
- ***Destination register***：20:16(rt, for lw, sw) / 15:11(rd, for R-type)

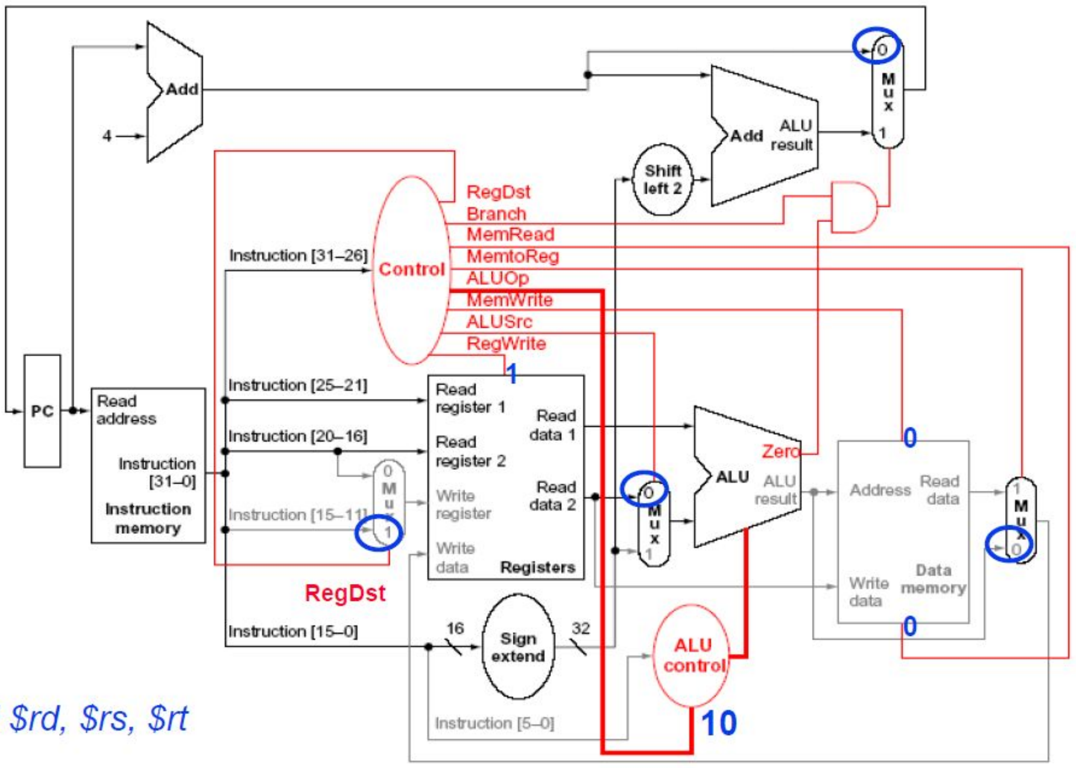




	0	1
RegDst	用到2個暫存器，讀取20:16的位置	用到3個暫存器，讀取15:11的位置
RegWrite	拒絕資料寫入	允許資料寫入
ALUSrc	使用data2出來的資料放入ALU	使用有號擴充後的資料放入ALU(通常是有關記憶體陣列)
PCSrc	PC=PC+4	使用跳躍指令(jump)時，如果要跳躍需要開啟
MemRead	拒絕記憶體資料讀取	允許記憶體資料讀取
MemWrite	拒絕記憶體資料寫入	允許記憶體資料寫入
MemtoReg	寫入暫存器的資料來自ALU算出(未使用記憶體)	寫入暫存器的資料來自記憶體

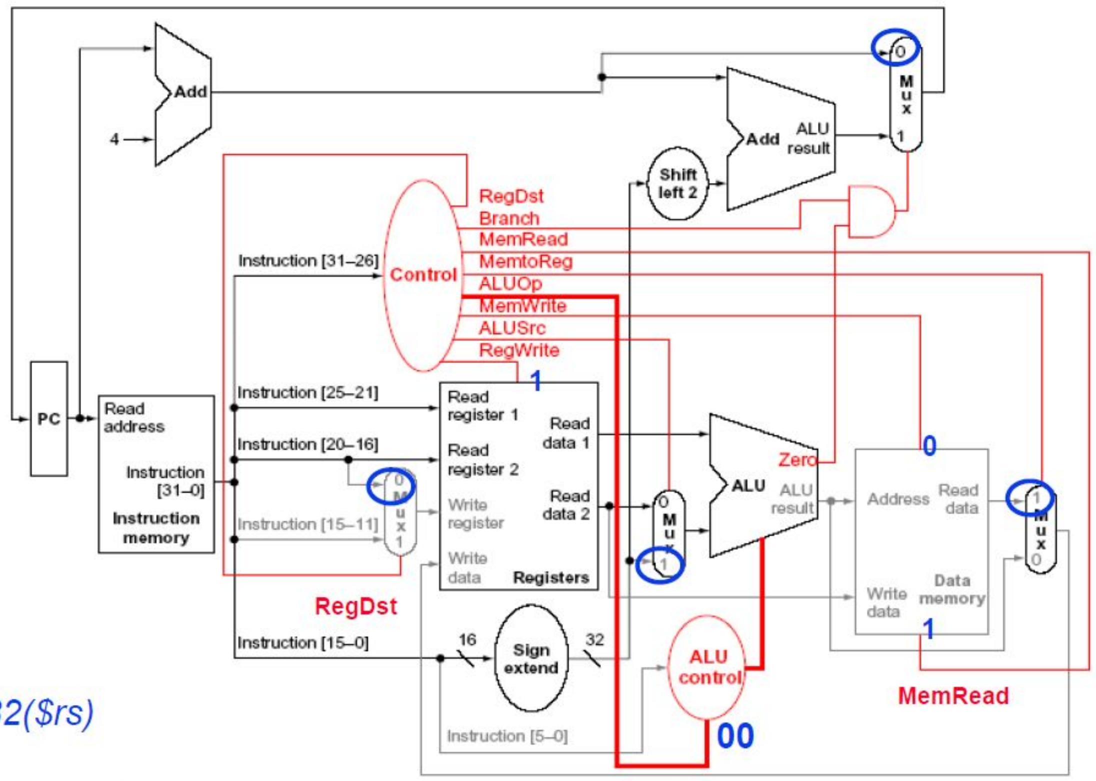


Signal name	Effect when deasserted (0)	Effect when asserted (1)
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.



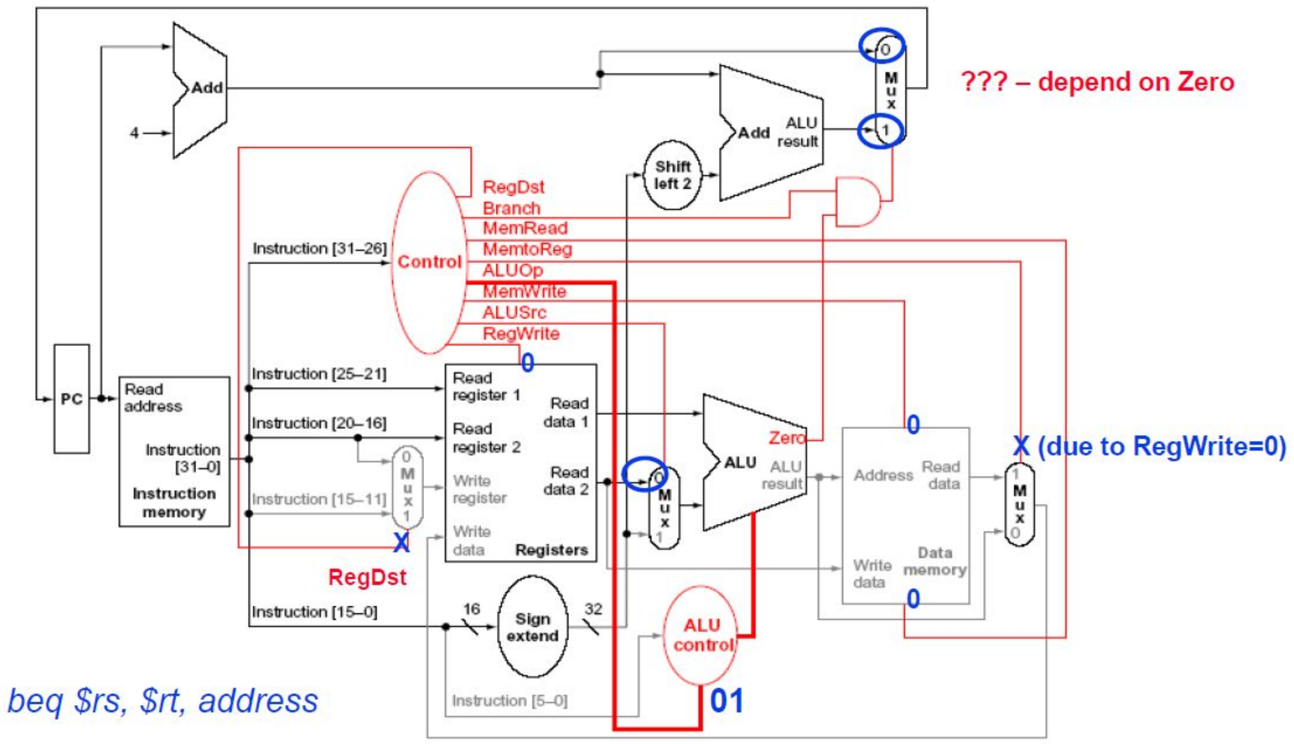
add $rd, $rs, $rt

欄位	0	rs	rt	rd	shamt	funct
位元位置	31:26	25:21	20:16	15:11	10:6	5:0



lw $rt, 32($rs)

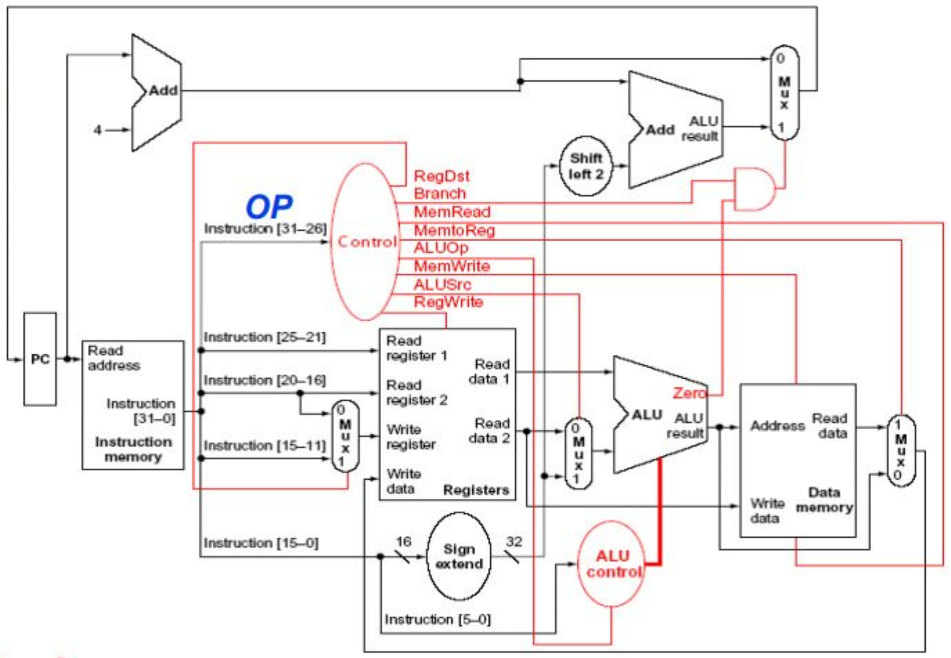
欄位	35 or 43	rs	rt	address
位元位置	31:26	25:21	20:16	15:0



beq $rs, $rt, address

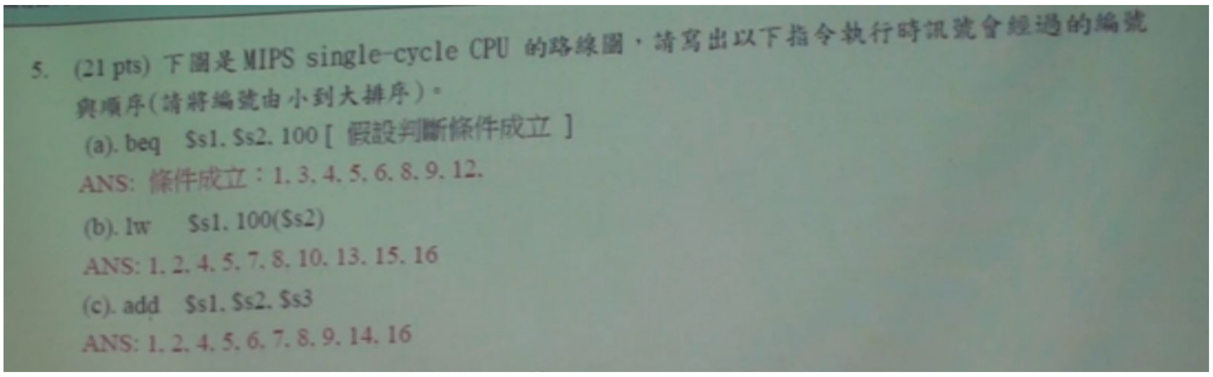
欄位	4	rs	rt	address
位元位置	31:26	25:21	20:16	15:0

Computer Org.



Main control

Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	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



5. (21 pts) 下圖是 MIPS single-cycle CPU 的路線圖，請寫出以下指令執行時訊號會經過的編號與順序(請將編號由小到大排序)。

(a). beq $s1. $s2. 100 [假設判斷條件成立]

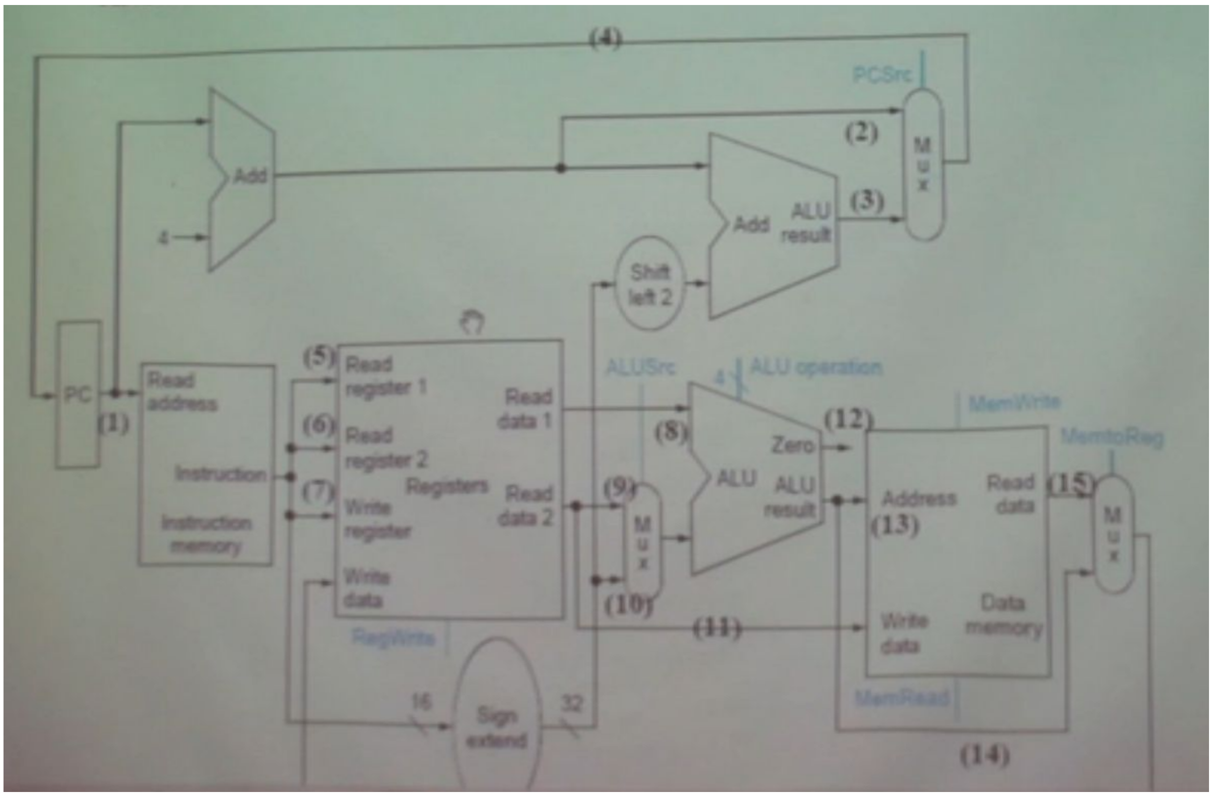
ANS: 條件成立：1. 3. 4. 5. 6. 8. 9. 12.

(b). lw $s1. 100($s2)

ANS: 1. 2. 4. 5. 7. 8. 10. 13. 15. 16

(c). add $s1. $s2. $s3

ANS: 1. 2. 4. 5. 6. 7. 8. 9. 14. 16



條件成立代表會進行跳躍

5、6、7，lw sw是5、7，因為它是兩個暫存器但是有使用到記憶體

像是beq bne使用兩個暫存器但是沒使用記憶體，所以是5、6

而add 使用了三個暫存器，所以是5、6、7

8、9、10，