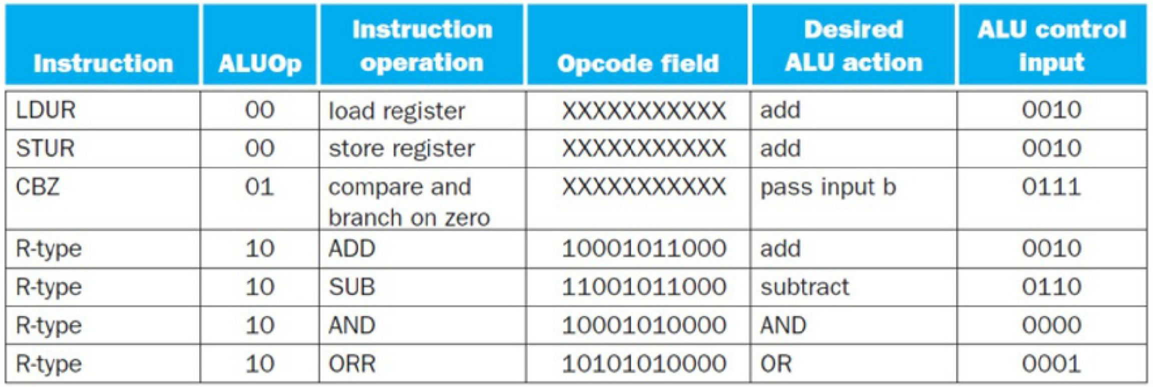
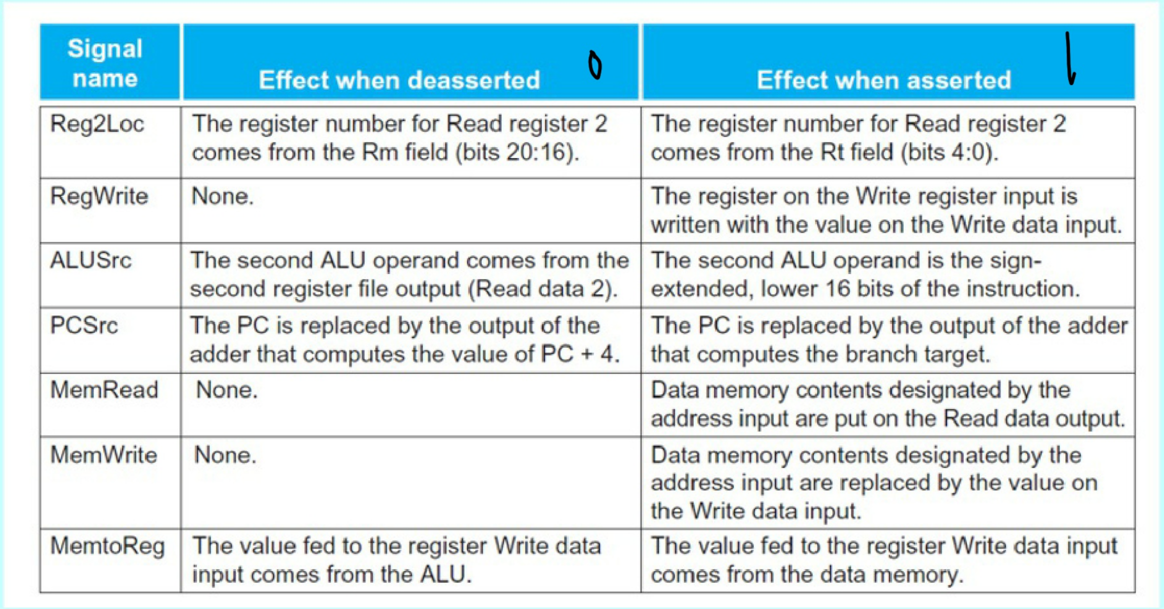
ALUSrc and ALUop

How ALU Control bits are set depends on the ALUop control bits



Instruction	ALUOp	Instruction operation	Opcode field	Desired ALU action	ALU control input
LDUR	00	load register	XXXXXXXXXXX	add	0010
STUR	00	store register	XXXXXXXXXXX	add	0010
CBZ	01	compare and branch on zero	XXXXXXXXXXX	pass input b	0111
R-type	10	ADD	10001011000	add	0010
R-type	10	SUB	11001011000	subtract	0110
R-type	10	AND	10001010000	AND	0000
R-type	10	ORR	10101010000	OR	0001

Control Signal



Signal name	Effect when deasserted 0	Effect when asserted 1
Reg2Loc	The register number for Read register 2 comes from the Rm field (bits 20:16).	The register number for Read register 2 comes from the Rt field (bits 4:0).
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 X4, X5, X6

Rd Rn Rm

X6 = X4 + X5

R - Format

opcode	Rm	shamt	Rn	Rd
11 bits	5 bits	6 bits	5 bits	5 bits

Instruction fields

- opcode: operation code
- Rm: the second register source operand
- shamt: shift amount (00000 for now)
- Rn: the first register source operand
- Rd: the register destination

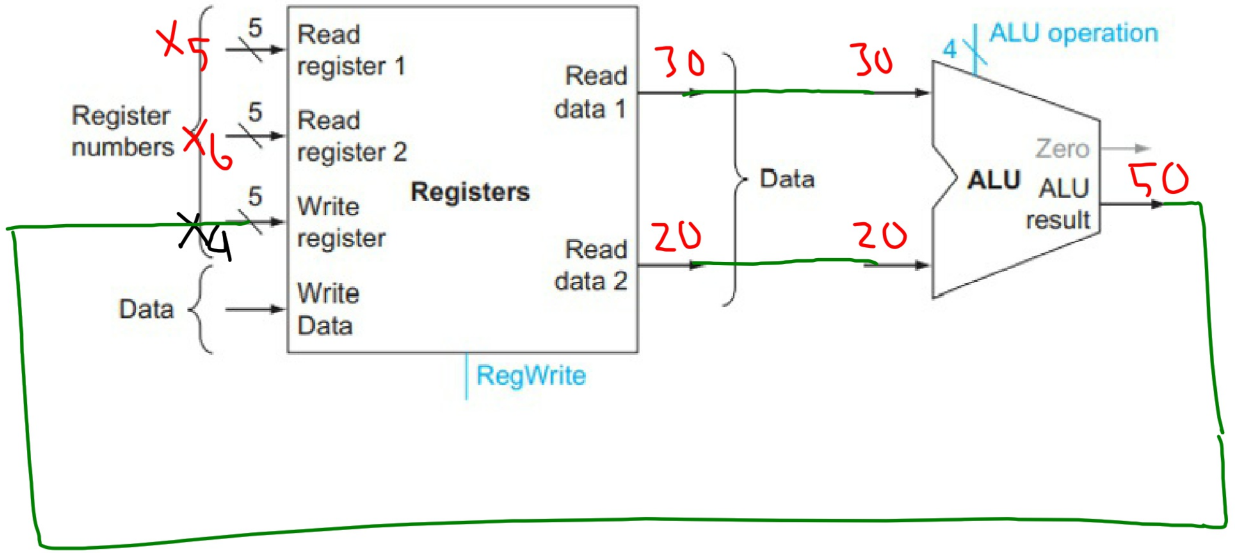
1112	6	0	5	4

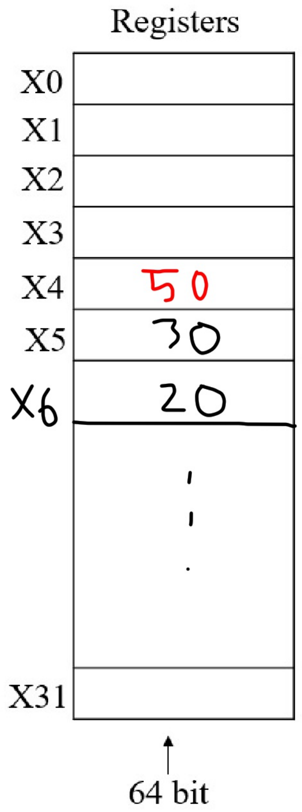
10001011000	00110	000000	01001	00100

← 32 bit machine code

Assume , X5 = 30 X6=20

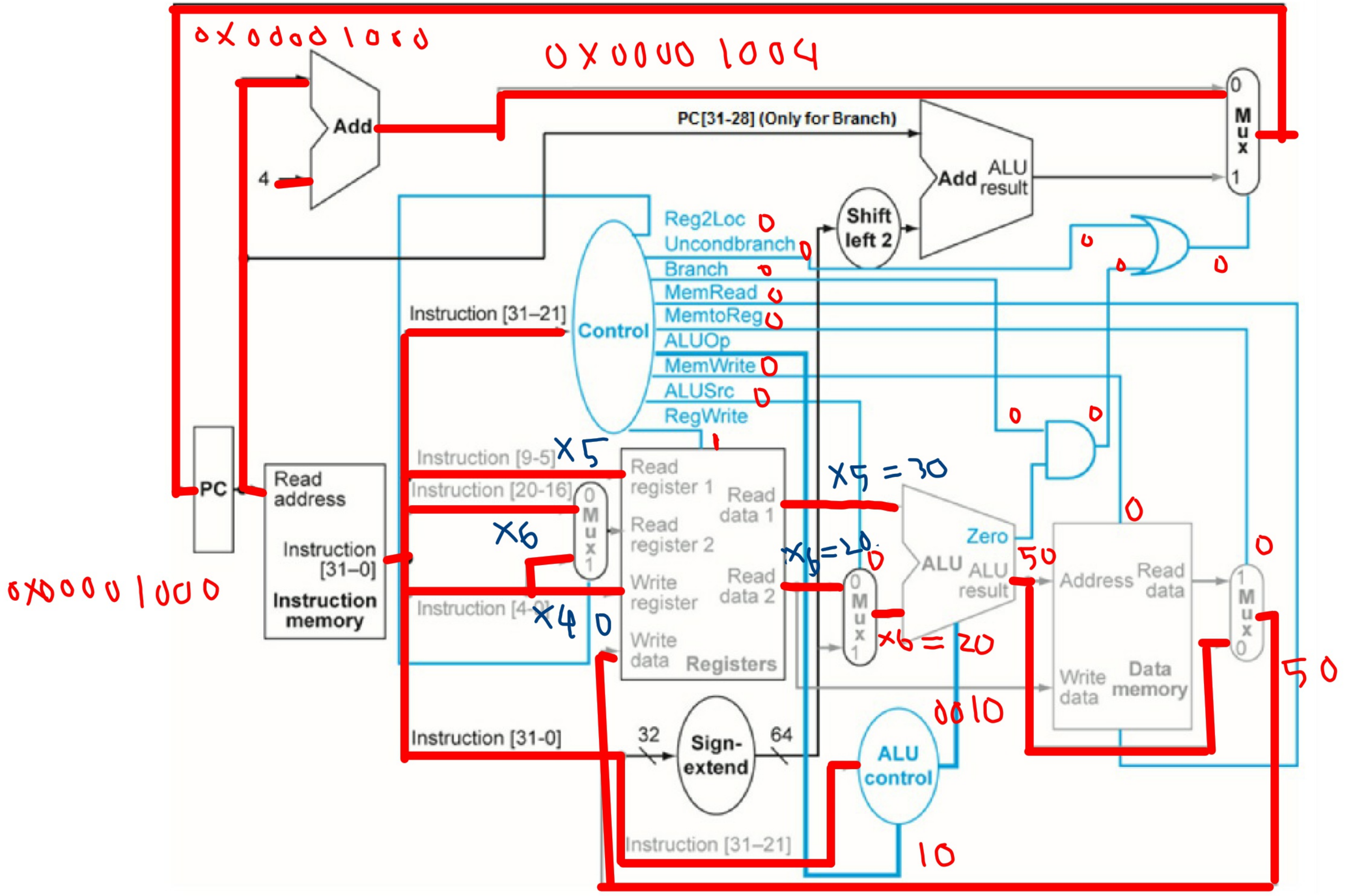
ADD X4, X5, X6





Tracing Datapath: ADD X4, X5, X6

→ R Type instruction



Reg2Loc = 0
Uncondbranch = 0
Branch = 0
MemRead = 0
MemtoReg = 0
ALUOp = 10
MemWrite = 0
ALUSrc = 0
RegWrite = 1

```
0X00001000        ADD X4, X5, X6

0X00001004
```