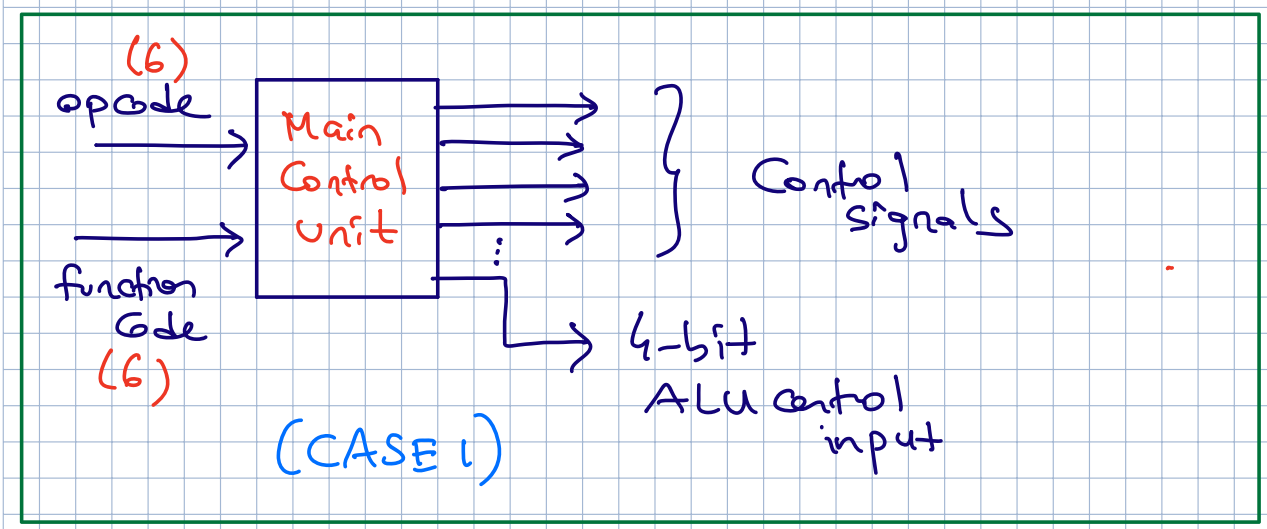
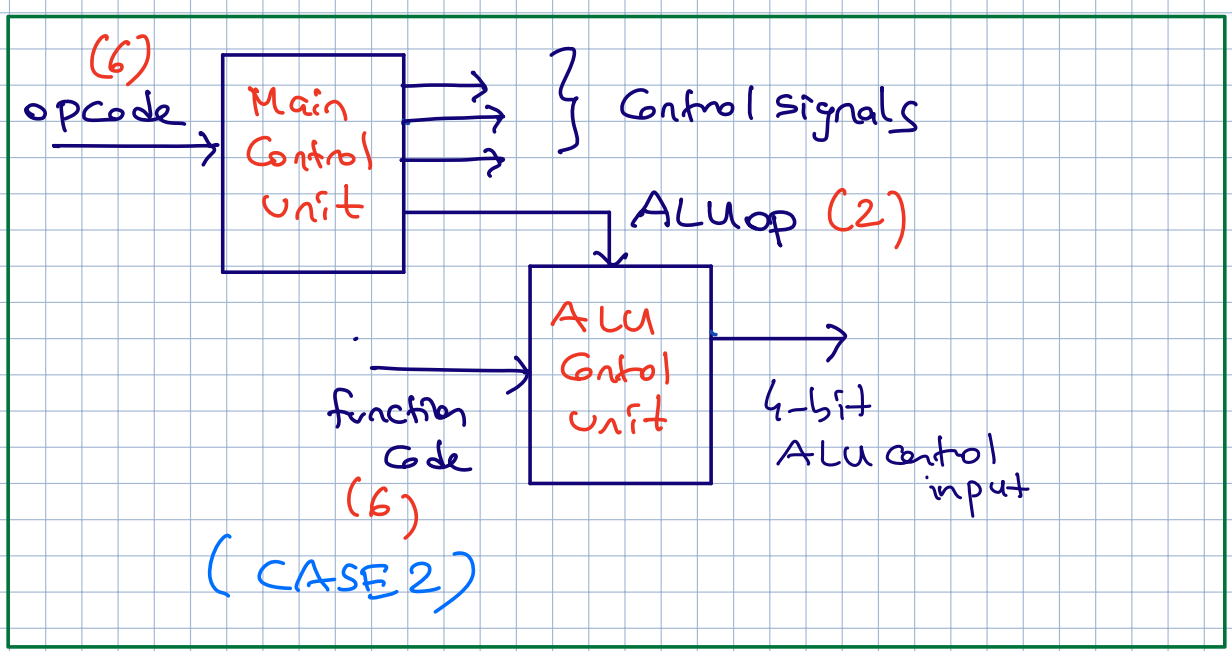
The ALU Control

Depending on Instruction class, ALU will need to perform

- for LW/SW : Compute memory addr. Calculation
- for R-type : add/sub/and/or/slt
- for branch : Subtraction





Why multiple level of control?

* Reduce size of main control unit
* Using several smaller control units increase speed of the control unit

Multiple Level of Control

4-bit ALU control input generated with a small unit that has inputs.

- function field of the instr.
- 2-bit control field (ALUop)

ALUop

- 00 → load/store (add)
- 01 → branch (subtract)
- 10 → R-type (determine operation encoded in function code)

Figure 4.12

(How ALU control bits are set deped on ALUop bits and different function codes for R-type instr.)

function codes: add = 32 Sub = 34
and = 36 or = 37 slt = 42

If ALUop = 00 or 01 ⇒ ALU action o not depend on function code (Do not care)

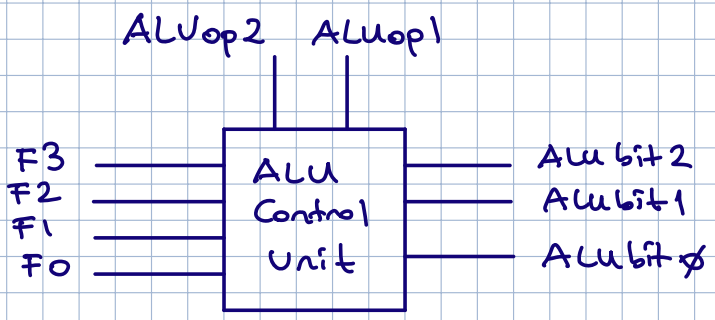
Figure 4.13

(The truth table for the 4 ALU control bits)

Inputs : ALUop & function code fields.

ALU does not use "11" encoding
(so truth table has 1X & X1 instead of 10 & 01)

F5 & F4 are always 10 ⇒ Don't care



MAIN CONTROL UNIT (OBSERVATIONS)

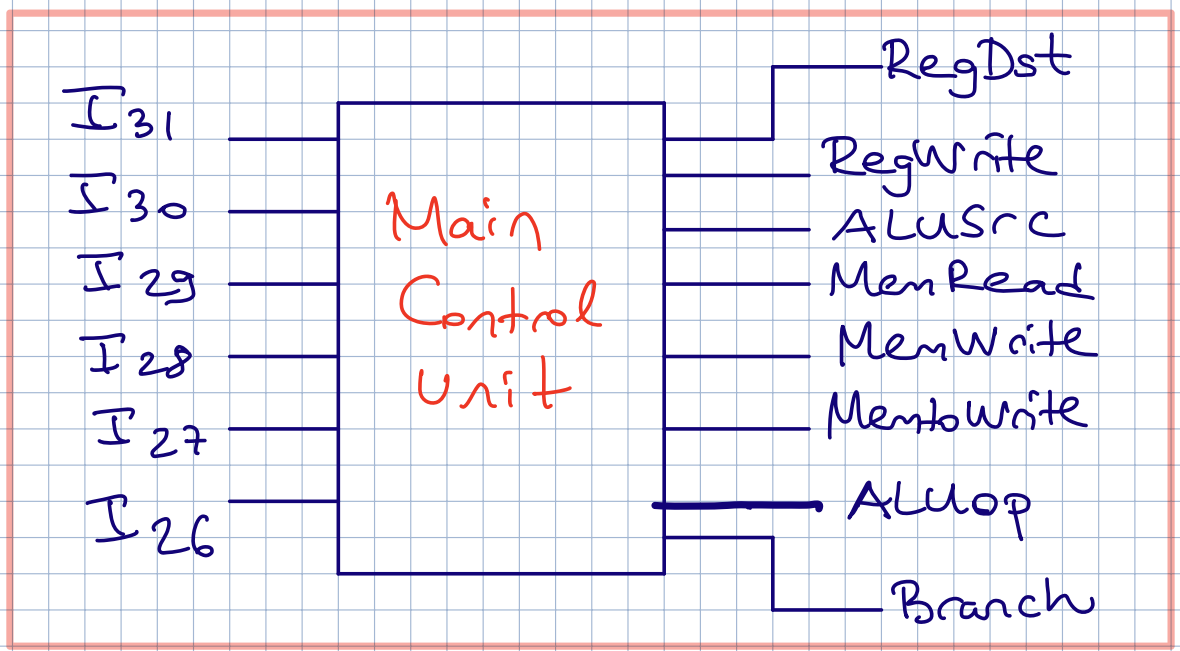
1. opcode is always in I[31:26]
2. Two registers to be read at positions I[25:21] & I[20:16] (for R-type, Branch, Store)
3. Base register for LW/SW: I[25-21]
4. 16-bit offset for branch equal, LW & SW → I[15:0]

(5) Destination register in two fields.

For LW ⇒ I[20:16]

For R-type ⇒ I[15:11]

(extra multiplexer required)

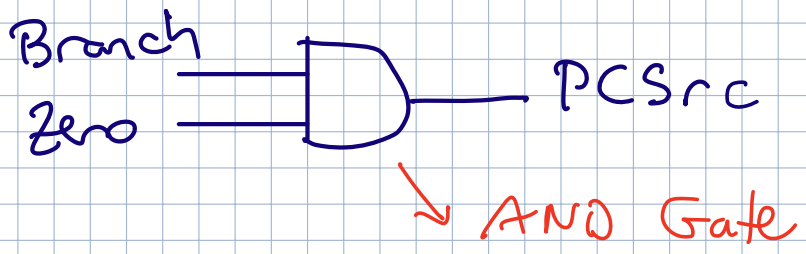


(7 control lines (1-bit) & 2-bit ALUop)

Main control unit set all control signals except PCSrc.

PCSrc is set if

* instr. is a branch
* Branch taken is true (Zero output of ALU is true)

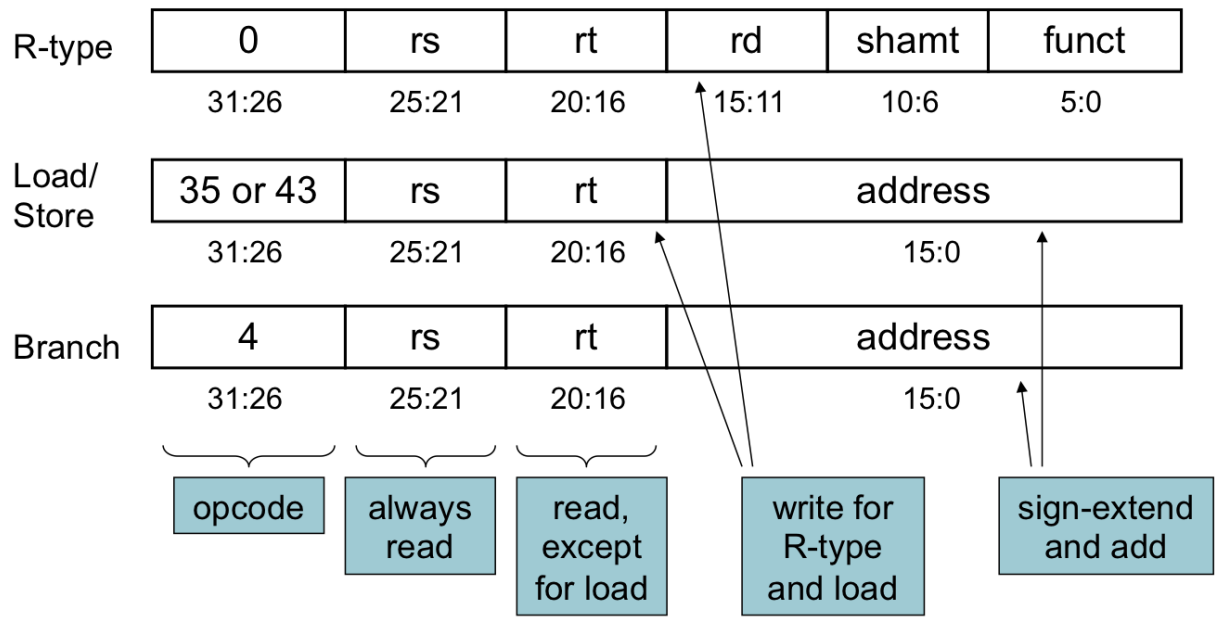


Finalizing the Control

opcode		Op5	Op4	Op3	Op2	Op1	Op0
0	R-format	0	0	0	0	0	0
35	LW	1	0	0	0	1	1
43	SW	1	0	1	0	1	1
4	Beq	0	0	0	1	0	0

Input signals that correspond to four opcodes that determine the control output settings.

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
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



R-type

add $t1, $t2, $t3

(1) Instr is feteled from instr. memory & PC is incremented

(2) $t2 & $t3 read from register file
MCU computes settings of control lines

(3) ALU operates on data read from register file using function code (I[5:0])

(4) Result from ALU is written into register file using bits [15:11] of instruction (select $t1)

Although everything is performed in 1 cc, we can think 4 steps to execute instr.

lw

lw $t1, 20($t2)

1. Instr is fetched from Instr. Memory & PC is incremented
2. $t2 value is read from register file
3. ALU performs the sum of value read from register file and sign-extended lower 16-bits of Instr.
4. ALU result is used as the addr for data memory.
5. Data from memory is written into register file

beq

beq $t1, $t2, offset

1. Instr fetched from memory and PC is incremented
2. $t1, $t2 read from register file
3. ALU perform a subtract on data values read from register file } Branch Decision

(PC+4) added to sign-extended lower 16-bits of instr. shifted left by 2 bits → Branch Target Addr.

(4) Zero output from ALU is used to decide which adder result to store in PC