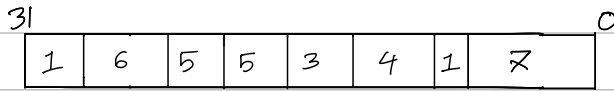


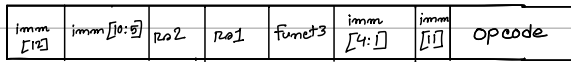
Branching Instructions

SB type instruction — beq, bne, blt, bge



* represents branch addresses in multiples of 2.

* each field has unique name and size.



* forward & backward moving.

* the unusual encoding of imm. simplifies datapath design.

Loop:

#80000 slli X10, X22, 3 ✓ — line 1

#80004 add X10, X10, X25 ✓ — line 2

#80008 ld X9, 0(X10) ✓ — line 3

#80012 bne X9, X24, Exit ✓ — line 4

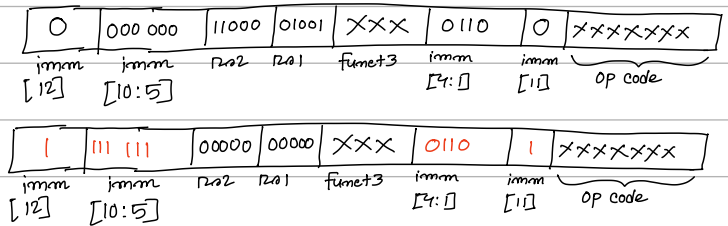
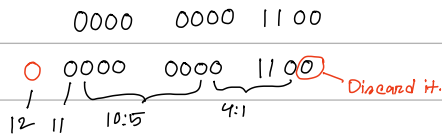
#80016 addi X22, X22, 1 ✓ — line 5 ✓

#80020 beq X9, X9, loop ✓ — line 6 ✓

Exit:

#80024 — line 7 ✓

$$3 \times 4 = 12$$



5 × 4 = 20 but its going upward so -20.

$$= 0000 \ 0001 \ 0100$$

$$= 0 \ 0000 \ 0001 \ 0100$$

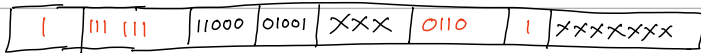
$$= 1 \ 111 \ 1110 \ 1011$$

PC relative addressing

$$= PC + immediate \times 2$$

$$+ 1$$

$$\begin{array}{ccccccc} 1 & 111 & 1110 & 1100 & 0 & 0 & 0 \\ 12 & 11 & 10:5 & 4:1 & 0 & 0 & 0 \end{array} \Rightarrow -20 \text{ in 2's com.}$$



0

(i) Form the 12 bit number

$$11 \ 111 \ 111 \ 0110$$

(ii) Detect if positive or negative numbers.

if neg perform 2's comp. again,

$$1111 \ 1111 \ 0110$$

$$0000 \ 0000 \ 1001$$

+1

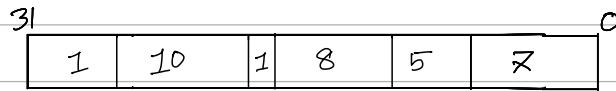
$$0000 \ 0000 \ 1010 \Rightarrow 10$$

(iii) PC (±) imm × 2

Based on the sign bit

offset

UJ type instruction - Jal



each field has unique name and size.



uses 20 bit immediates for larger jumps.

For more large jump, (32 bit)

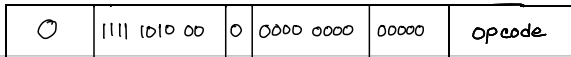
lui: load address [31:12]

to a temp reg.

jalr: add address [11:0] and

jump to target.

Jal x0, 2000



2000 = 0000 0000 0111 1101 0000

= 0 0000 0000 0111 1101 0000
 20 10:12 11 10:1

Given a branch on register x10 being equal to zero,

beq x10, x0, L1

replace it by a pair of instructions that offers a much greater branching distance.

Answer

These instructions replace the short-address conditional branch:

bne x10, x0, L2

jal x0, L1

L2: