

la \$t1, array

.data
a: .word 10 ← 1001 0000
array: .word 20, 30, 40 ← 1001 00 04
b: .word 50 ← 1001 00 10

la \$t1, array
[lui \$at, 0x1001
ori \$t1, \$at, 0x0004

\$at = \$1

1001	0000
0000	0004
or	
1001	0004

\$t1

address of array
in \$t1

lw

lw \$t2, b
lui \$at, 0x1001
lw \$t2, 0x0010(\$at) ←

\$at

1001	0000
------	------

1001 0010 ← address of b
load from this
address to \$t2

beq : branch forward

Address (hex)

... 20

24

28

2C

30

beq \$t0, \$t3, next
add \$t0, \$t0, \$t0 } 1
add \$t0, \$t0, \$t0 } 2
add \$t0, \$t0, \$t0 } 3
next: add \$t1, \$t1, \$t1

beq \$t0, \$t3, next

beq \$t0, \$t3, next ← I type

I type imm ← size in bits

6	5	5	16
op	rs	rt	imm
4max	8	11	3

00 0100 01000 01011 0000 0000 0000 0011
1 1 0 B 0 0 0 2

⇒ 0x110B0003 ← machine code

beg: branch backward

Address

0x00400024	next2: add \$t0, \$t0, \$t0	4
28	add \$t0, \$t0, \$t0	3
2c	add \$t0, \$t0, \$t0	2
30	beg \$t0, \$t1, next2	1
34	add \$t0, \$t0, \$t0	

;

bin

$-4 \Rightarrow$ in 8 bits

dec	bin
4	0000 0100
	↓ invert
	1111 1011
	↓
+	1111 1100

beg \$t0, \$t1, next2

beg \$8, \$9, next2 ← size in bits

6	5	5	16
opcode	rs	rt	imm

4 8 9 -4 ← use its 2's complement

00 0100 01000 01001 1111 1111 1111 1100

1 1 0 9 F F F C

machine code

0X11 09 FF FC

Pseudo branch instructions blt, bgt, ble

For their implementation set less than and beq or bne is used.
(set) + (beq | bne) \Rightarrow pseudo branch instructions

bgt \$t0, \$t1, next
≡ blt \$t1, \$t0, next

[set \$at, \$t1, \$t0
 bne \$at, \$zero, next]

bge \$t0, \$t1, next
≫ ≡ not <
branch if \$t0 not < \$t1

set \$at, \$t0, \$t1
bgt \$at, \$zero, next

length 0x00400034

jump instruction

Diagram illustrating the instruction format (32 bits total):

- 6 bits: opcode
- 26 bits: operand
 - 2 bits: rex (labeled "2 rex")
 - 24 bits: remaining operand bits

Annotations:

- ← no. of bits (pointing to the 26-bit operand field)
- does not include the first 4 bits of PC (pointing to the 24-bit field)
- does not include last two bits (pointing to the 24-bit field)

0040 0024 j next
- 28 add - -
- 2C add - -
- 30 add - -
0040 0034 next: add - -

[illegible]

Given 0x08100007 what is this instruction?

PC
0x00400034
0000

0x08100007
0000 1000 0001 0000 0000 0000 0000 0111 ← 26 bits

jump

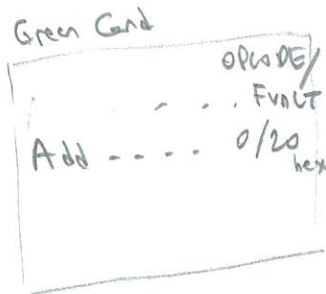
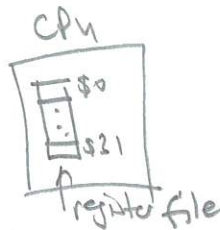
0000 0001 0000 0000 0000 0000 0111 00
0 0 4 0 0 0 1 C

add 2's

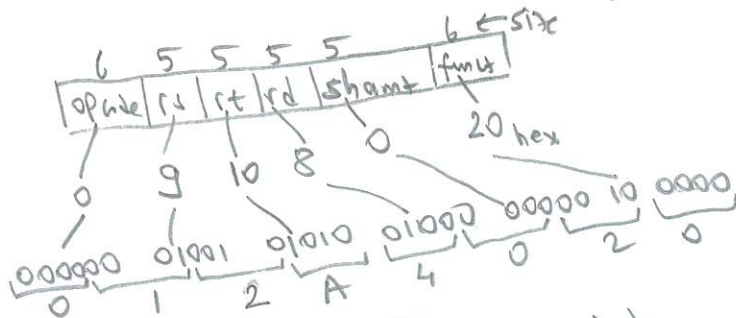
It branches to mem. location 0x0040001c

Assembling Instructions

rd rs rt
add \$t0, \$t1, \$t2
(8) (9) (10)
 $R[rd] = R[rs] + R[rt]$



\$t0 = \$8
\$t1 = \$9
\$t2 = \$10



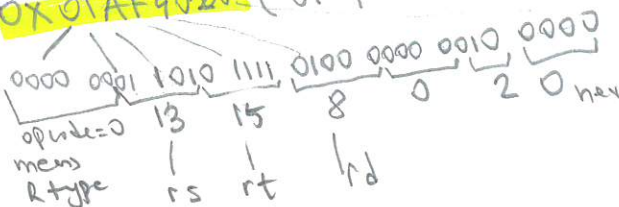
machine int
0x012A4020

add \$t0, \$t1, \$t2 → Assembler → 0x012A4020
Assembling like compiling
← Disassembling

Disassembling Instructions

①

0x01AF4020 (01af4020)



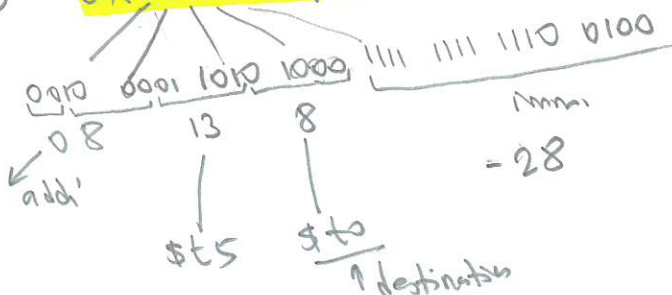
add rd, rs, rt
add \$8, \$13, \$15
add \$t0, \$t5, \$t7

0x01AF4020 $\xRightarrow{\text{disas.}}$ add \$t0, \$t5, \$t7

\$t0 = \$8
\$t1 = \$9
\$t2 = \$10
\$t3 = \$11
\$t4 = \$12
\$t5 = \$13
\$t6 = \$14
\$t7 = \$15

②

0x21A8FFEA



addi \$t0, \$t5, -28

$R[rt] = R[rs] + \text{SignExt Imm}$
addi rt, rs, imm

