CS224
Class Notes
Oct. 12, 2020

Assembling la, lw
Code Generation for bdt, jump
Disassembling instructions

Oct. 12, 2020

la

la $t1, arrey

```
    .data
a:  .word  10          <- 1001 0000
array: .word 20,30,40  <- 1001 0004
b:  .word  50          <- 1001 0010
```

```
la  $t1, array
lui $at, 0x1001
ori $t1, $at, 0x0004
```

$at = $1 | 1001 | 0000

or | 0000 | 0004

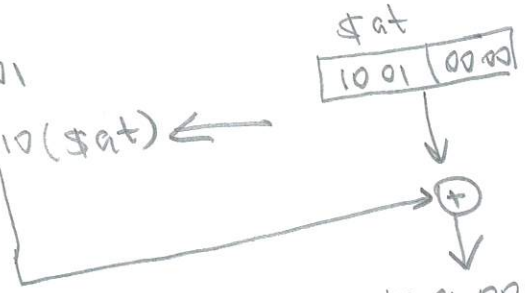
$t1 | 1001 | 0004

address of array in $t1

---

lw

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



---

beq : branch forward

Address (hex)		
... 20		beq $t0, $t3, next
24		add $t0, $t0, $t0
28		add $t0, $t0, $t0
2c		add $t0, $t0, $t0
30	next:	add $t1, $t1, $t1

1, 2, 3

beq $t0, $t3, next

beq $8, $11, next ← I type

I type inst ← size in bits

op code	rs	rt	imm
6	5	5	16
4 hex	8	11	3

00 0100 01000 01011 0000 0000 0000 0011

1 1 0 B 0 0 0 3

⇒ 0x1108 0003 ← machine code

beq : branch backward

```
Address
0x00400024  next2: add $t0, $t0, $t0
          28       add $t0, $t0, $t0
          2c       add $t0, $t0, $t0
          30       beq $t0, $t1, next2
          34       add $t0, $t0, $t0
           :
```

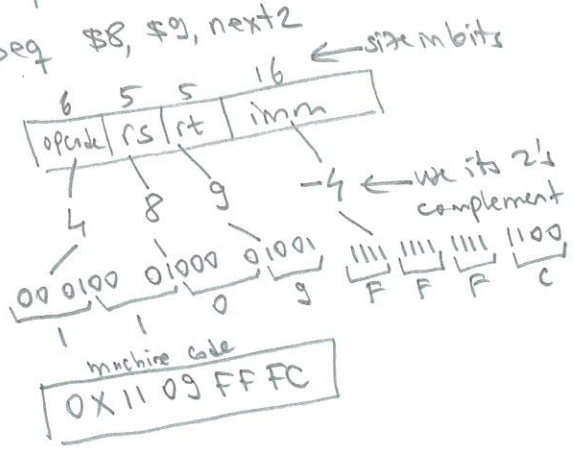
4 → 3 → 2 → 1

-4 ⇒ in 8 bits

dec	bin
4 =	0000 0100
↓ invert	
	1111 1011
+	1
	1111 1100

beq $t0, $t1, next2

beq $8, $9, next2



Pseudo branch instructions blt, bgt, ble

For their implementation set less than and beq or bne is used.

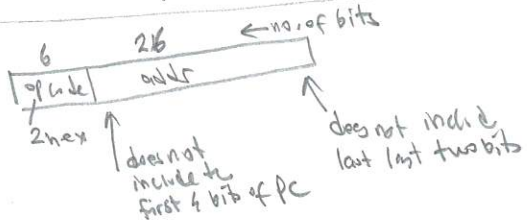
(slt) + (beq | bne) ⇒ pseudo branch instructions

```
blt $t0, $t1, next
    slt $at, $t0, $t1
    bne $at, $zero, next
```

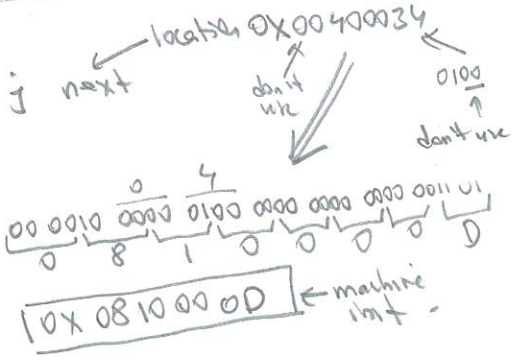
```
bgt $t0, $t1, next
≡ blt $t1, $t0, next
    slt $at, $t1, $t0
    bne $at, $zero, next
```

```
bge $t0, $t1, next
≥ ≡ not <
branch if $t0 not < $t1
    slt $at, $t0, $t1
    beq $at, $0, next
```

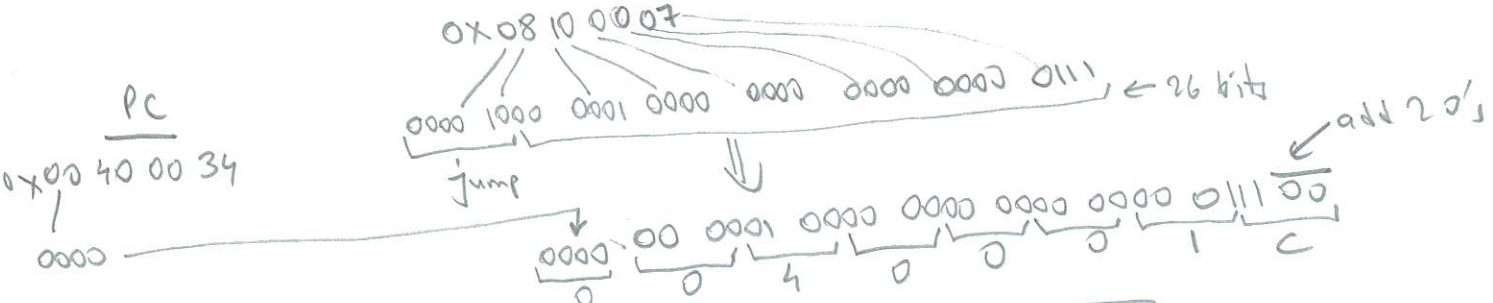
jump instruction




```
00400024    j next
      28    add . .
      2c    add . .
      30    add . .
00400034 next: add . .
```



Given 0X08100007 what is this instruction?



It branches to mem. location 0X0040001C

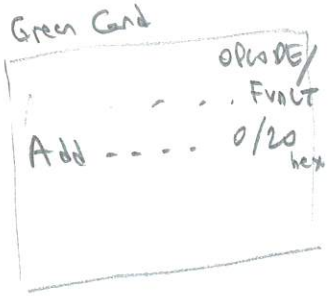
Oct. 12, '20

Assembling Instructions

CPU

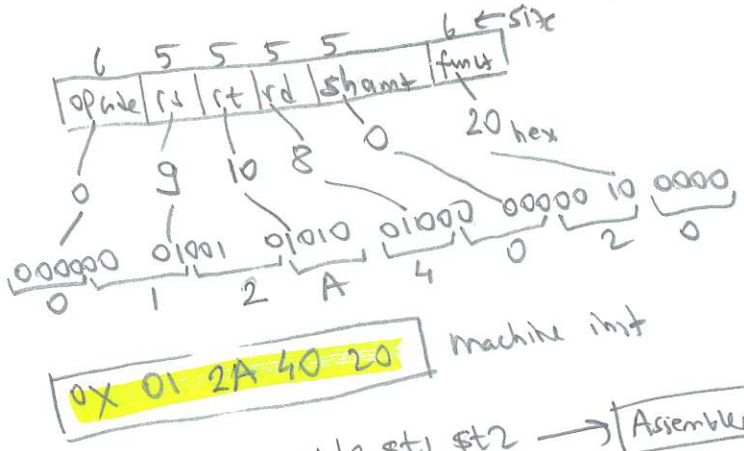
register file

Green Card



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

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



0X 01 2A 40 20 machine inst

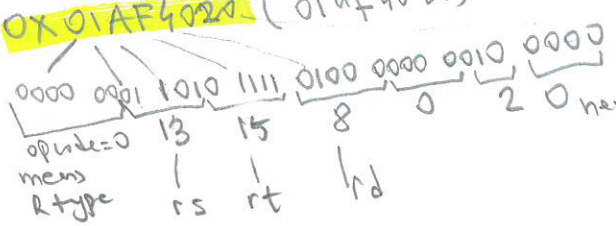
add $t0, $t1, $t2 → Assembler → 0X 01 2A 40 20

Assembling like compiling

← Dissassembling

Dissassembling Instructions

① 0X 01AF4020 (01af4020)



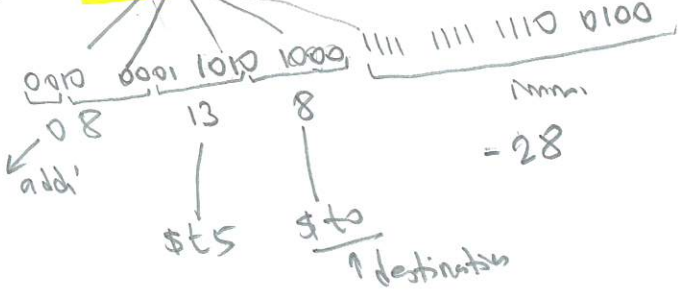
structure
opcode/rs/rt/rd/shamt/funct

0X 01 AF 40 20 ⟹ dissas. ⟹ add $t0, $t5, $t7

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

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

② 0X 21A8FFE4



addi $t0, $t5, -28

R[rt] = R[rs] + SignExtImm

addi rt, rs, Imm