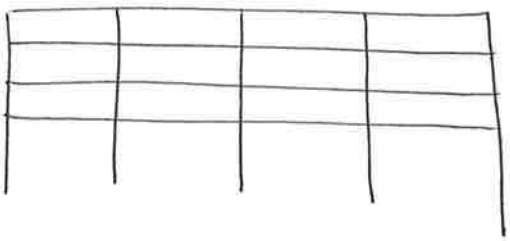
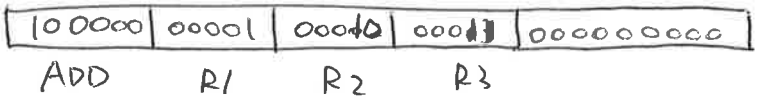
S9 ☐ The assembly language

- The assembly language

- code



write instructions directly by checking OPCODE

e.g. 

- assembly provides an easier way to write code

  just write ADD(R2, R3, R1)
  in text file.

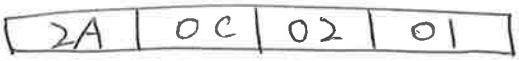
- how is it done

write byte by byte, translate

1 2 0xC

0b101010

...



...

left to right
by space

lower

- define ~~var~~ macro X= 123 X
- specify location

  . ↔ current address

  labels

text

translation (binary)

```
1   2   0xC
0b101010
X=123        } macro   7B
X
. = 0x8
1   2   3   4
. = . +3
```

2A	0C	02	01	00000000
			7B	00000004
04	03	02	01	00000008
01				
		0F	10	0000000C
				00000010

```
lab1  1
. align → to next word
16 lab1
```

If some addr skipped, filled with 0.
The default start 0

- Allow expressions

  +, −, *, /, %, <<, >>

text

translation

```
1   2   3+3
5x6
R0=0
R1=1
 :
```

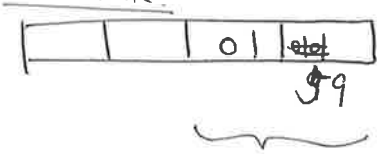
1E	06	02	01
		04	02

R31=31 → special register, constant 0, talk in beta

```
. -2
R1 <<2
```

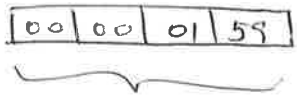
- function macros

.macro WORD(X) X%256 (X/256)%256

macro
WORD(345) ↔ translation 89 1 ↔ translate 

1x256+5x16+9=345

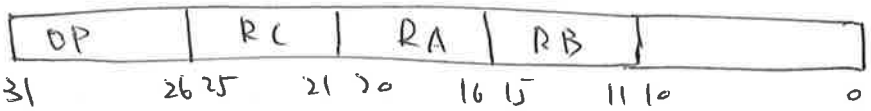
.macro LONG(x) WORD(x) WORD(x>>16)

macro
LONG(345) ↔ translation WORD(345) WORD(0) ↔ 89 1 0 0 ↔ translate 

32 size.

.macro betaop(OP, RA, RB, RC)

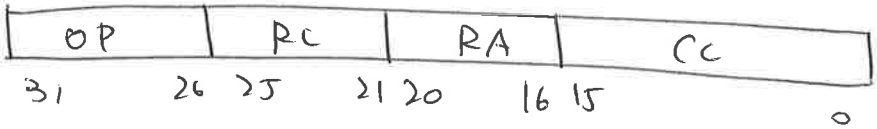
LONG((OP<<26)+(RC<<21)+(RA<<16)+(RB<<11))



ADD

.macro betaopc(OP, RA, CC, RC)

LONG((OP<<26)+(RC<<21)+(RA<<16)+(CC%0x10000))



ADDC ... LD. ST JMP

.macro betabr(OP, RA, RC, CC)

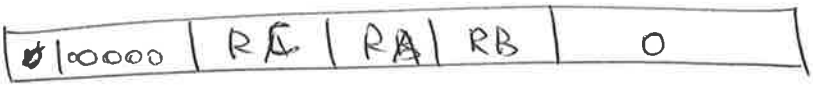
betaopc(OP, RA, ((CC-(.+4))>>2), RC)

CC in betaopc X (why?)

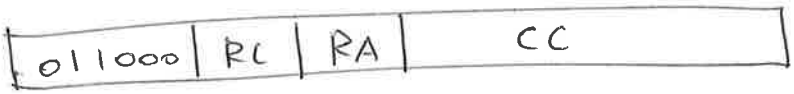
$X = (CC - (PC+4)) >> 2$

$CC = PC+4+4*X$. (LABEL) of jump.

. macro ADD(RA, RB, RC) betaop(0x20, RA, RB, RC)



. macro LD(RA, CC, RC) betaopc(0x18, RA, CC, RC)



. macro ST(RC, CC, RA) betaopc(0x19, RA, CC, RC)

. macro BEQ(RA, LABEL, RC) betabr(0x1D, RA, RC, LABEL)

all the instructions now can write.

- Extending the functionality

. macro MOVE(RA, RC) ADD(RA, R31, RC)

. macro CMOVE(CC, RC) ADDC(R31, C, RC)

. macro BF(RA, LABEL, RC) BEQ(RA, LABEL, RC)
↳ false

. macro BT(RA, LABEL, RC) BNE(RA, LABEL, RC)

. macro BR(LABEL, RC) BEQ(R31, LABEL, RC)

. macro LD(CC, RC)

. macro ST(RC, CC)

. macro COM(RA, RC) XORC(RA, -1, RC)
↳ ~RA

. macro NEG(RA, RC) SUB(R31, RA, RC)

. macro NOP() ADD(R31, R31, R31)

HALT ⟶ LONG(0)

ADD 35 , 99

```
    LD (X, R1)
    LD (y, R2)
    ADD (R1, R2, R0)
    ST (R0, Z)
    HALT
X:  LONG (35)
y:  LONG (99)
Z:  LONG (0)
```