

8.1 Assemblers

An **assembler** is a program that converts assembly instructions into machine instructions (0's and 1's). The assembler's tasks are:

1. Replacing pseudoinstructions with native instructions
2. Determining each label's memory address
3. Generating machine instructions for the assembly instructions

Replacing pseudoinstructions

A **pseudoinstruction** is an assembly instruction that must be replaced by one or more native instructions before being executed. The assembler does such replacement.

PARTICIPATION ACTIVITY

8.1.1: An assembler's first task is replacing pseudoinstructions by native instructions.

Start ☐ 2x speed

```
# Goal: $t0 = |$t1| + $t2
blt $t1, $zero, Negate
add $t0, $t1, $zero
j Cont
```

```
Negate: addi $t3, $zero, -1
        mul $t0, $t3, $t1
```

```
Cont:   add $t0, $t0, $t2
```

```
# Goal: $t0 = |$t1| + $t2
slt $t3, $t1, $zero
bne $t3, $zero, Negate
add $t0, $t1, $zero
j Cont
```

```
Negate: addi $t3, $zero, -1
        mult $t3, $t1
        mflo $t0
```

```
Cont:   add $t0, $t0, $t2
```

**PARTICIPATION
ACTIVITY**

8.1.2: Replacing pseudoinstructions by native instructions.



Refer to the animation above.

1) blt is replaced by how many instructions?



☐ 1

☐ 2

2) mul is replaced by how many instructions?



☐ 1

☐ 2

3) The program with pseudoinstructions has 6 instructions. The program with native instructions has how many instructions?



☐ 6

☐ 8

Determining label addresses

Assembly instructions use labels for branches/jumps, but machine instructions use numerical offsets or addresses. Thus, assembler is to create a **symbol table**, which lists an address for each label.

**PARTICIPATION
ACTIVITY**

8.1.3: An assembler's second task is to determine label addresses, kept in a symbol table.



Symbol	Address
--------	---------

Start — 2x speed

Negate	16
Cont	28

```
# Goal: $t0 = |$t1| + $t2
0 slt $t3, $t1, $zero
4 bne $t3, $zero, Negate
8 add $t0, $t1, $zero
12 j Cont
```

```
Negate: 16 addi $t3, $zero, -1
        20 mult $t3, $t1
        24 mflo $t0
```

```
Cont:   28 add $t0, $t0, $t2
```

**PARTICIPATION
ACTIVITY**

8.1.4: Label addresses: Symbol table.

Consider the following assembly. Assume this portion of the program will be placed in instruction memory starting at address 200.

```
# Branch example
    bne $t0, $t1, Else1
    addi $t3, $t3, 50
    j Cont
```

```
Else1: bne $t0, $t2, Else2
        addi $t3, $t3, 60
        j Cont
```

```
Else2: addi $t3, $t3, 70
```

```
Cont:
    ...
```

1) What is the address of `bne $t0, $t1, Else1`?

Check

Show answer

2) What is the address of `addi $t3, $t3, 50`?

Check

[Show answer](#)

3) What is the address of label `Else1`?

Check

[Show answer](#)

4) What is the address of label `Else2`?

Check

[Show answer](#)

5) What is the address of label `Cont`?

Check

[Show answer](#)

6) For the shown program, how many symbols will be in the symbol table?

Check

[Show answer](#)

7) When examining the program from top to bottom, is the address of label `Else1`

known at the first instruction `bne $t0, $t1, Else1`? Type yes or no.

Check

Show answer

8) When examining the program from top to bottom, is the address of label `Cont` known at the third instruction `j Cont`? Type yes or no.

Check

Show answer

Generating machine instructions

After pseudoinstructions have been replaced by native instructions, and all label addresses are determined, each native instruction is translated to a machine instruction.

PARTICIPATION ACTIVITY

8.1.5: An assembler replaces pseudoinstructions by native ones, puts determined label addresses in a symbol table, and finally generates machine instructions.

Start

☐ 2x speed

Symbol	Address	Binary	Immediate
Neg	16	10000	0..0100
Cont	28	11100	0000..111

```
# $t0 = |$t1| + $t2
blt $t1, $zero, Neg
add $t0, $t1, $zero
j Cont
```

```
# $t0 = |$t1| + $t2
0 slt $t3, $t1, $zero
4 bne $t3, $zero, Neg
8 add $t0, $t1, $zero
12 j Cont
```

```
0 000000 01001 00000 01011 00000 101010
4 000101 01001 01010 0000000000000100
8 000000 01001 00000 01001 00000 100000
12 000010 000000000000000000000000111
```

Neg:	addi \$t3, \$zero, -1	Neg:	16 addi \$t3, \$zero, -1	16	0001000	000000	01001	00000100000010110000
	mul \$t0, \$t3, \$t1		20 mult \$t3, \$t1	24	000000	000000	00000	01000 00000 010010
			24 mflo \$t0	28	000000	01000	01010	01000 00000 100000
Cont:		Cont:						
	add \$t0, \$t0, \$t2		28 add \$t0, \$t0, \$t2					

Assembly w/ pseudoinstructions
Assembly w/ native instructions
Machine instructions

Above, the spaces in the machine instructions do not really exist, and are shown for readability only.

PARTICIPATION ACTIVITY

8.1.6: Assembler.

Consider the animation above.

- 1) The assembler replaced the mul pseudoinstruction by two native instructions.

□

☐ True
☐ False
- 2) The assembler determined the instruction memory address of labels Neg and Cont.

□

☐ True
☐ False
- 3) Given a program consisting of native assembly instructions and a symbol table, the assembler could convert each

□

native assembly instruction to a machine instruction one at a time.

- ☐ True
- ☐ False

4) A reasonable alternative approach is to generate the symbol table before replacing pseudoinstructions by native instructions, using each pseudoinstruction's address.

- ☐ True
- ☐ False

5) The above assembler made three passes over assembly: One to replace pseudoinstructions, one to create a symbol table, and one to convert each assembly instruction to a machine instruction.

- ☐ True
- ☐ False

**PARTICIPATION
ACTIVITY**

8.1.7: Machine instructions.

Assembly

```

Line 1 # t3 = M[5000] * M[5004]
Line 2 lw $t1, 0($t0)
Line 3 lw $t2, 4($t0)
Line 4 mul $t3, $t1, $t2
Line 5
Line 6 # Loop twice
Line 7 addi $t4, $zero, 0
Line 8 addi $t5, $t5, 2
Line 9 loop:
Line 10     beq $t4, $t5, store
Line 11     addi $t4, $t4, 1
Line 12     mul $t3, $t3, $t3
Line 13     j loop
Line 14
Line 15 store:
Line 16 sw $t3, 8($t0)

```

Machine instructions

Instruction memory

Text

Bits

ASSEMBLE
ENTER SIMULATION
STEP
RUN

More options ▼

 [Provide feedback on this section](#)