

Lecture 7

Loading

CS 241: Foundations of Sequential Programs
Fall 2014

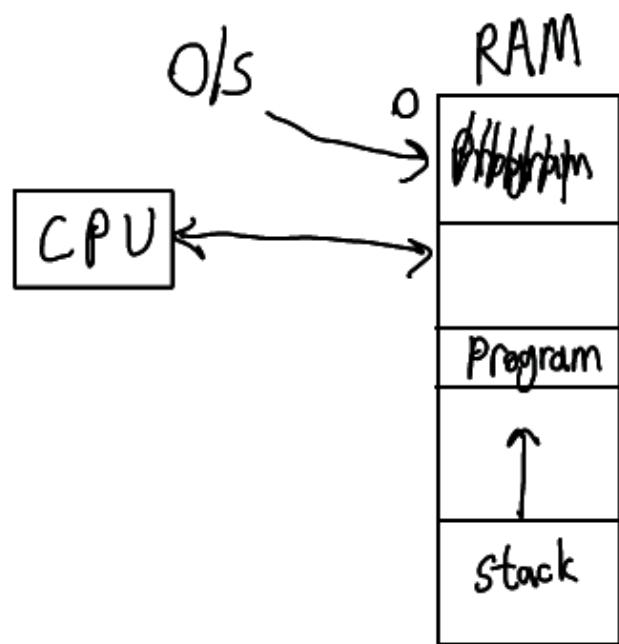
Troy Vasiga et al
University of Waterloo

Correctness

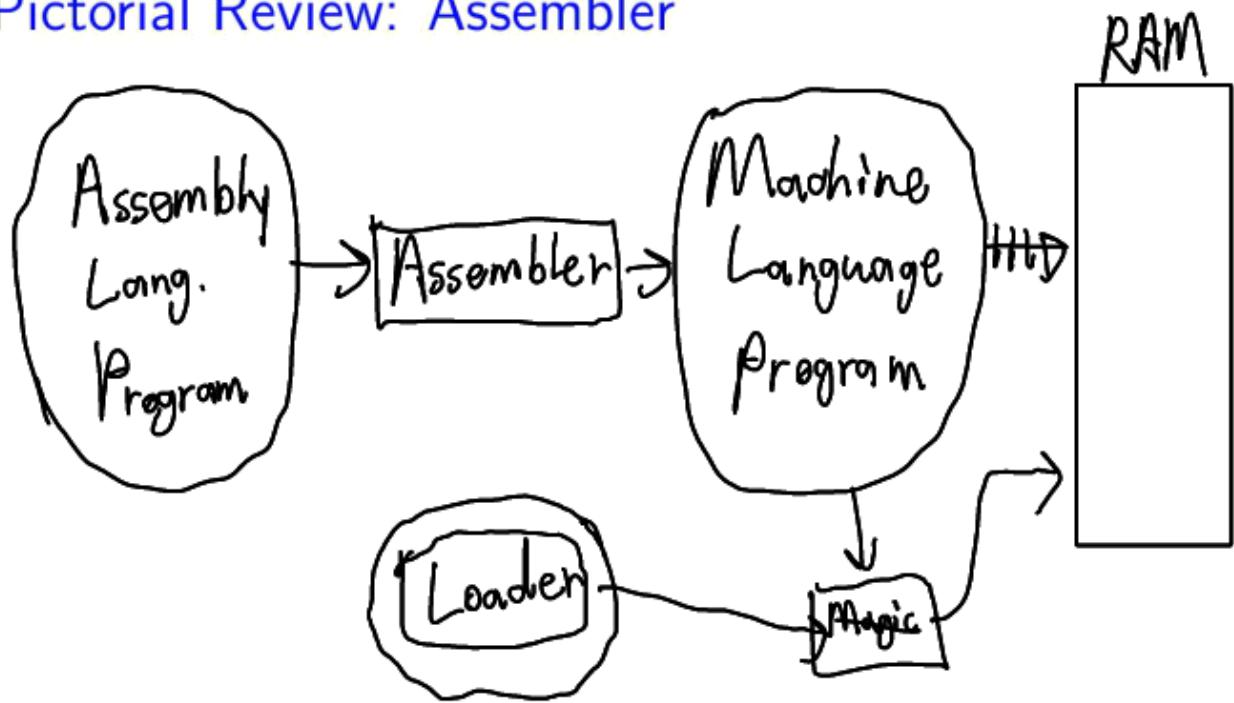
- ▶ **Read the spec carefully!**
- ▶ think reasonably and unreasonably
- ▶ remembering memory
- ▶ running times do matter
- ▶ **Don't use Marmoset as your only testing tool!**

RTFM

Pictorial Review: CPU and RAM

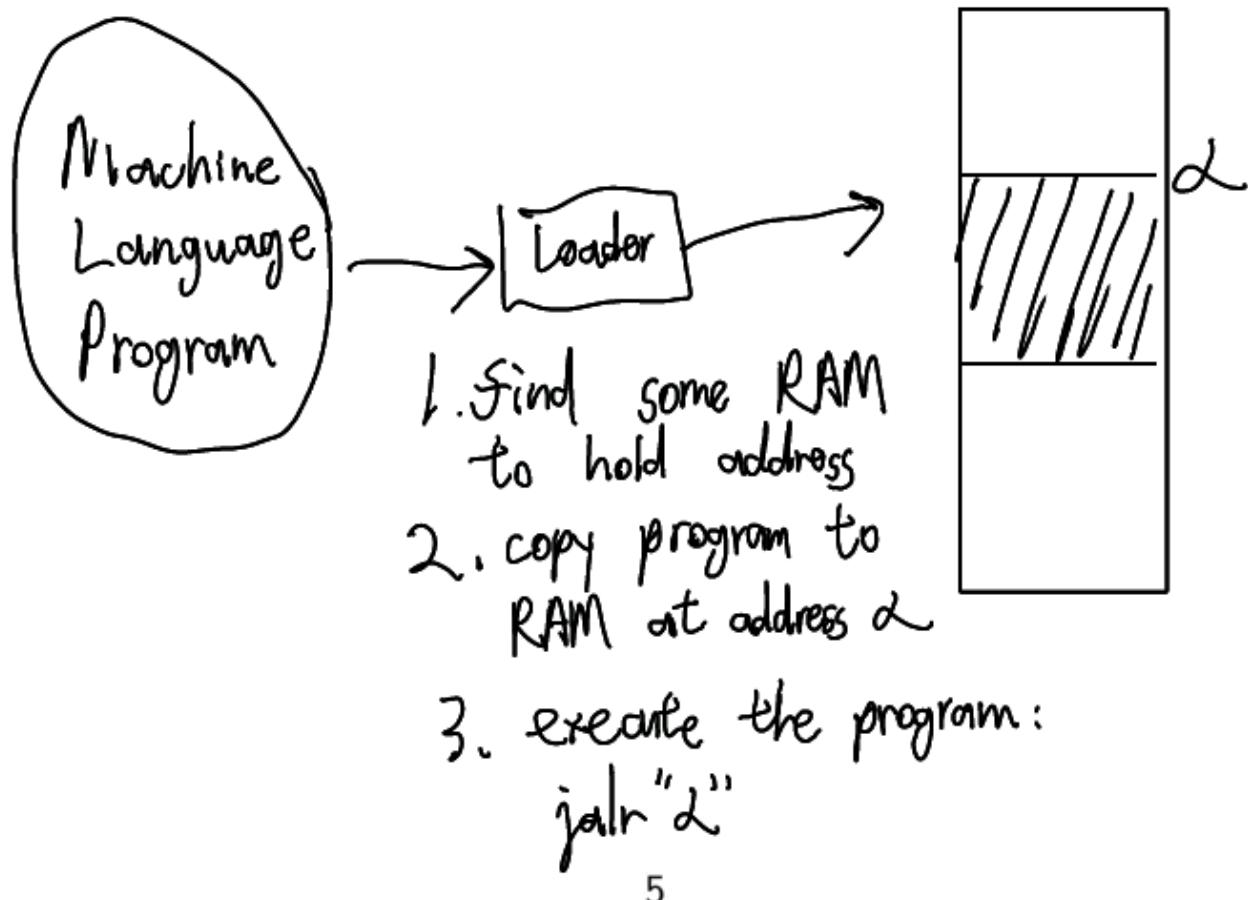


Pictorial Review: Assembler



The magic of loaders

This is what the operating system does!



Loader pseudocode

Version 0.1

```
loop
    decide which program to run ] CS 350 / OS course
    figure out length of the program (n) ] find space
    find n words of storage at address  $\alpha$  ]  

    read program into memory at  $\alpha$  ] copy program
    set up program (e.g., two ints) ]  

    put  $\alpha$  into a register (say $29) ] execute
    jalr $29
endloop
```

Chickens and eggs

- ▶ What is the size of the program we want to load?

[To determine size, load into RAM]
[To load into RAM, determine size]

- ▶ Isn't the loader also a program?

↳ microloader / Boot loader

BIOS "firmware"

"Boot strapping"

Example Program

```
0 lis $3
4 .word fortytwo
8 lw $3, 0($3)
c jr $31
10 fortytwo:
     .word 42
```

Symbol Table	
0	fortytwo
4	
8	
c	
10	

RAM	
0	0x00001814
4	0x00000010
8	-----
c	-----
10	0x0000002A

Why does this work? When does it not work?

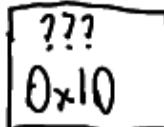
↳ base address is 0.

Notice how mips.twoints works:

```
@ubuntu1204-004[101]% java mips.twoints
Usage: java mips.twoints <filename> [load_address]
```

Example Program Again

```
lis $3  
.word fortytwo  
lw $3, 0($3)  
jr $31  
fortytwo:  
.word 42
```

\$3 

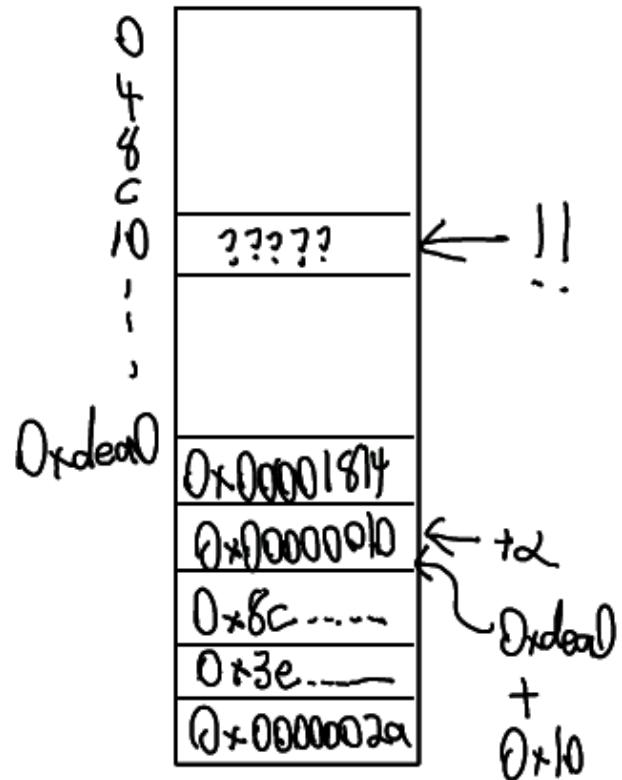
What breaks here? How can we fix it?

.word <label> ↪ absolute address!!

What else breaks in MIPS?

↪ beq/bne?

↪ relative address
∴ fine_g

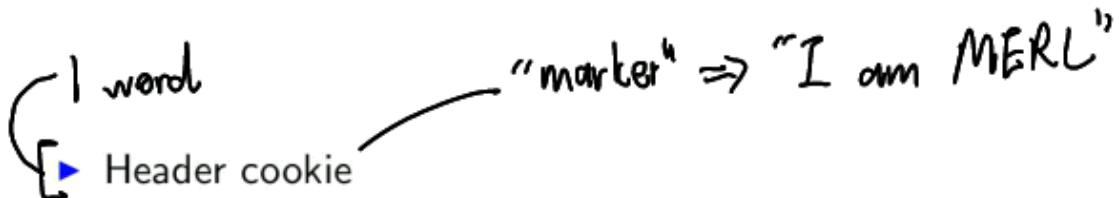


Solving this problem

MIPS
Executable
Relocatable
Linkable
Format

MERL

MERL Format



1 word [▶ Length of entire .merl file

1 word [▶ Length of just MIPS program

n words [▶ Program code

▶ Notes of what to change

See full description on CS241 webpage.

relocation
entries: each entry looks like
entry type —→ .word |
addressing —→ .word location
to fix

Relocating Loader Pseudocode

```

read header
 $\alpha = \text{findFreeRAM}(\text{codeLength})$  } find space
for each instruction
     $\text{MEM}[\alpha+i] = \text{instruction}$ 
for each relocation entry
     $\text{MEM}[\alpha+\text{location}] += \alpha$ 
place  $\alpha$  into $29
jalr $29

```

version 1.0

copy into
RAM at
address α
fix
subtract c

RAM

0	0x10000002	Updated due to MEL header
4	0x0000 0028	
8	0x0000 0014	
C	0x0000 1814	
10	0x0000001c	
14	0x8c ----	
18	0x3e ----	
1C	0x00000029	
	0x00000001	← relocate
	0x00000010	← addressing to fix

MIPS Program

A MERL Assembler

- ▶ Pass 1 changes:

- ↳ keep track of length
 - ↳ when I encounter .word <label>
 - ⇒ record a relocation entry with the current location
 - ↳ start counting address at 0xc

- ▶ Pass 2 changes:

- ↳ output header
 - ↳ output notes