中山大學 SUN YAT-SEN UNIVERSITY

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

Compiler Design
编译器构造实验

Lab 13: Project 4

张献伟

xianweiz.github.io

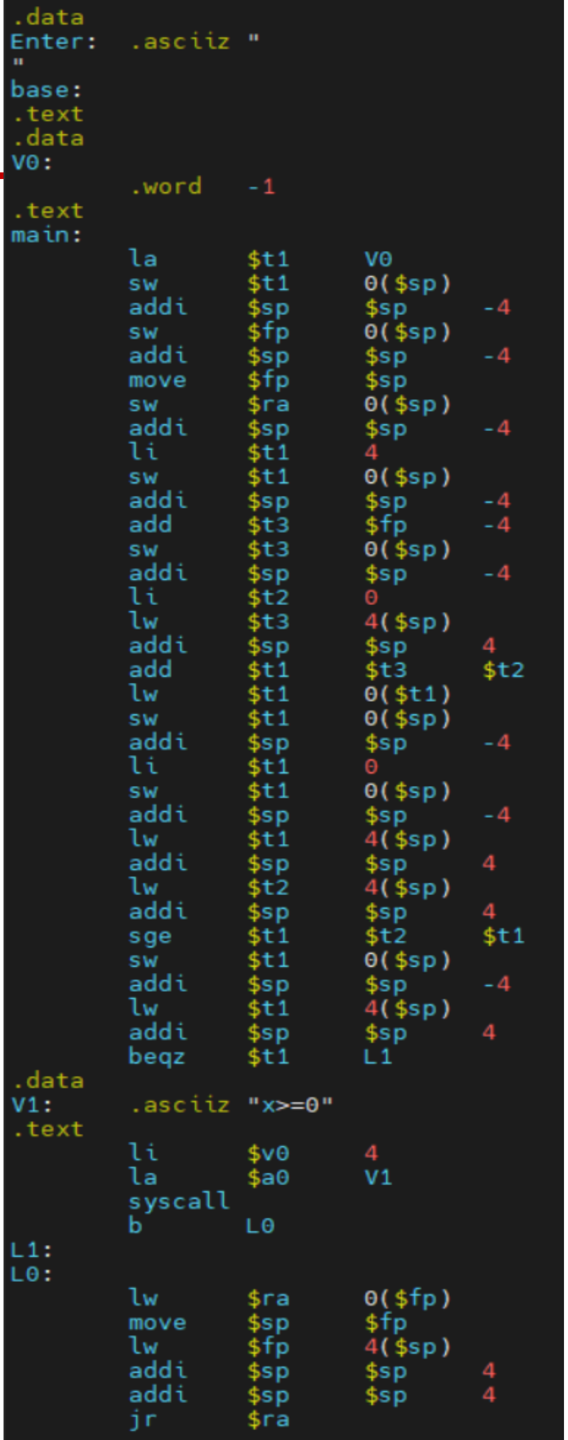
DCS292, 6/3/2021

Overview

- Take as input the augmented AST and symbol table to produce assembly code
  - Add offset attributes to your symbol table entries to calculate the position of each variable in memory
  - Translate subtrees of the AST into sequences of assembly instructions
    - Traverse AST to get the statements of source code
    - Then translate the statements into instructions
- Test the generated R2000/3000 assembly code using SPIM
  - SPIM reads MIPS assembly instructions from a file and executes them on a virtual MIPS machine

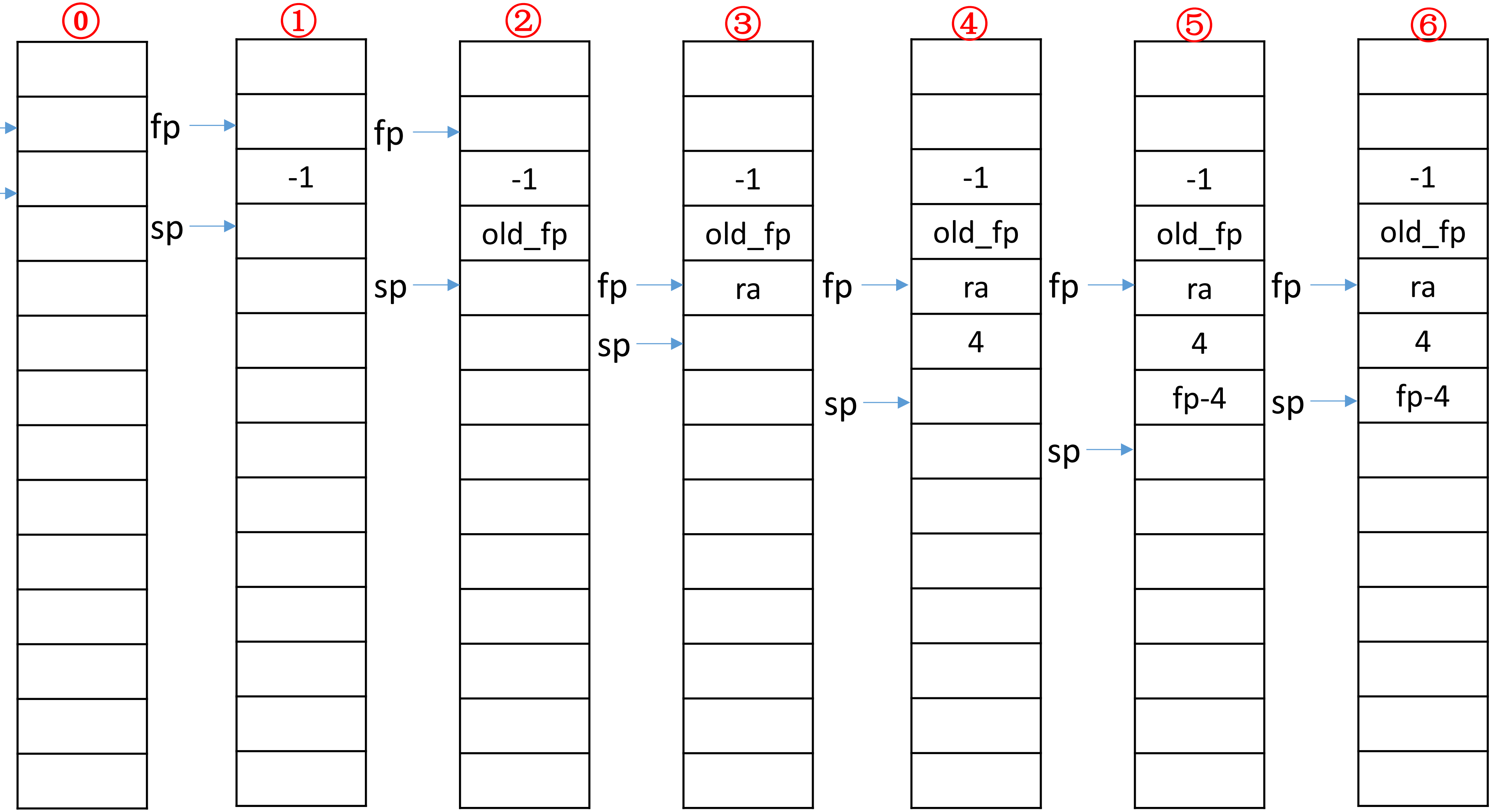
Example (proj4/tests/src1.s)

```
/* Ex1: Assignment statement */
program ex1;
class c1
{
   declarations
      int x=-1;
   enddeclarations
   method void main()
   declarations
      int x=4;
   enddeclarations
   {
     if (x>=0)
     {
       system.println('x>=0');
     };
   }
}
```



```
.data
Enter:  .asciiz "
"
base:
.text
.data
V0:
        .word   -1
.text
main:
        la      $t1     V0
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        sw      $fp     0($sp)
        addi    $sp     $sp     -4
        move    $fp     $sp
        sw      $ra     0($sp)
        addi    $sp     $sp     -4
        li      $t1     4
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        add     $t3     $fp     -4
        sw      $t3     0($sp)
        addi    $sp     $sp     -4
        li      $t2     0
        lw      $t3     4($sp)
        addi    $sp     $sp     4
        add     $t1     $t3     $t2
        lw      $t1     0($t1)
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        li      $t1     0
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        lw      $t1     4($sp)
        addi    $sp     $sp     4
        lw      $t2     4($sp)
        addi    $sp     $sp     4
        sge     $t1     $t2     $t1
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        lw      $t1     4($sp)
        addi    $sp     $sp     4
        beqz    $t1     L1
.data
V1:     .asciiz "x>=0"
.text
        li      $v0     4
        la      $a0     V1
        syscall
        b       L0
L1:
L0:
        lw      $ra     0($fp)
        move    $sp     $fp
        lw      $fp     4($sp)
        addi    $sp     $sp     4
        addi    $sp     $sp     4
        jr      $ra
```

src1.s (proj4/docs/src1_commented.s)



```
sw $t1 0($fp)
addi $sp $sp -4
```

```
sw $fp 0($fp)
addi $sp $sp -4
```

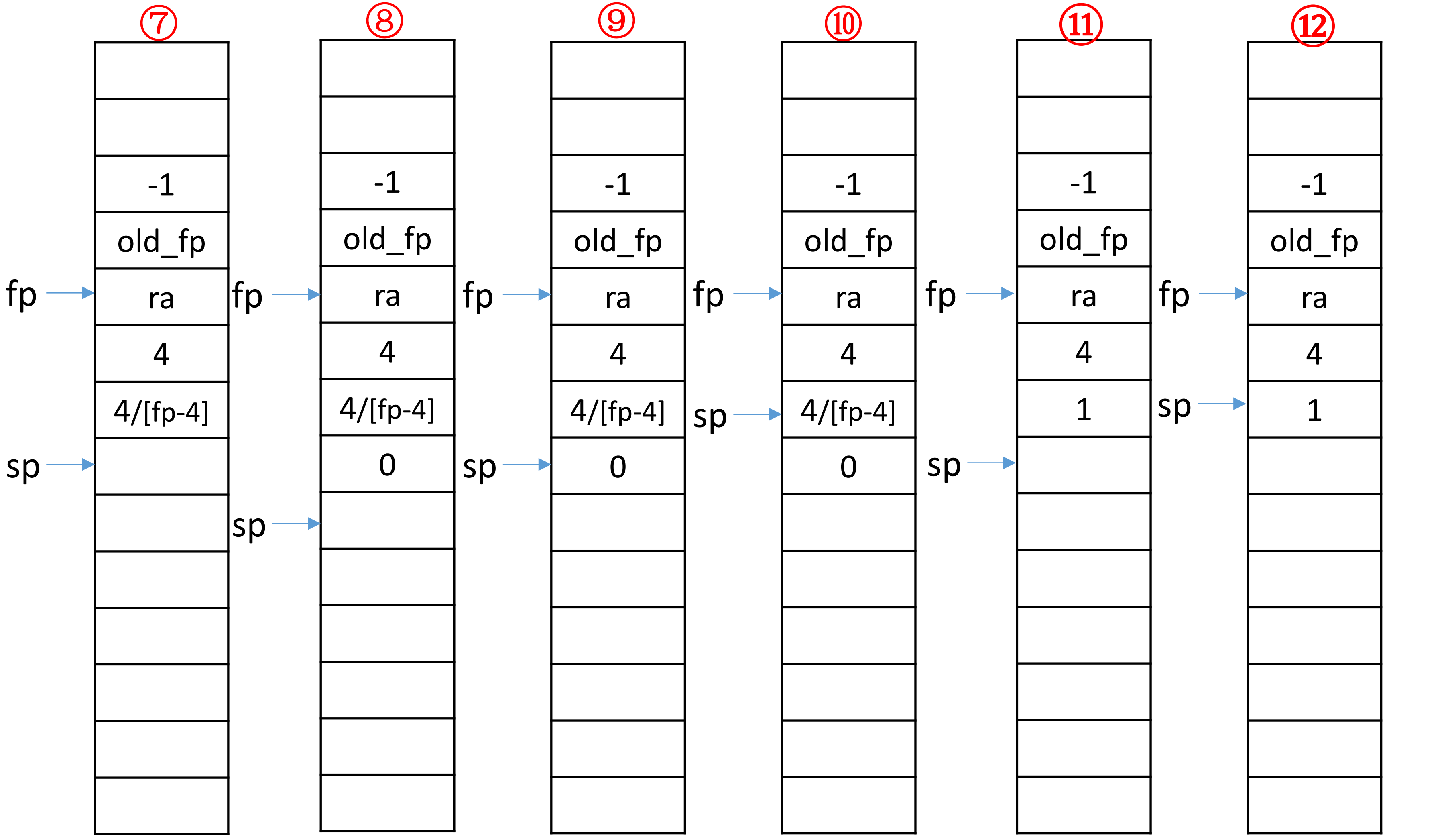
```
move $fp $sp
sw $ra 0($sp)
addi $sp $sp -4
```

```
li $t1 4
sw $t1 0($sp)
addi $sp $sp -4
```

```
add $t3 $fp -4
sw $t3 0($sp)
addi $sp $sp -4
```

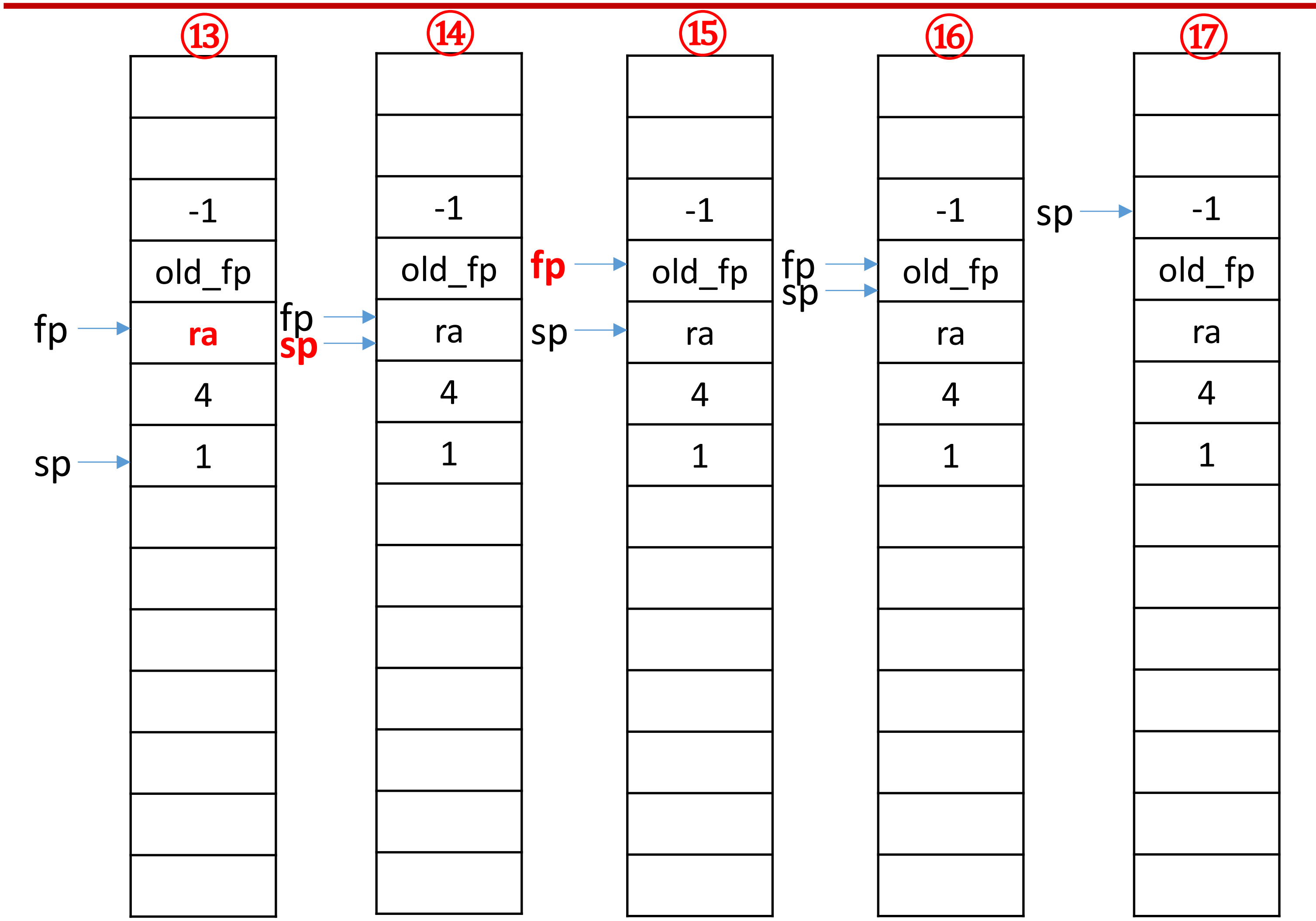
```
add $t2 0
sw $t3 4($sp)
addi $sp $sp 4
```

src1.s (proj4/docs/src1_commented.s)



⑦	⑧	⑨	⑩	⑪	⑫
-1	-1	-1	-1	-1	-1
old_fp	old_fp	old_fp	old_fp	old_fp	old_fp
fp → ra	fp → ra	fp → ra	fp → ra	fp → ra	fp → ra
4	4	4	4	4	4
4/[fp-4]	4/[fp-4]	4/[fp-4]	sp → 4/[fp-4]	1	sp → 1
sp →	0	sp → 0	0	sp →	
	sp →				
add $t1 $t3 $t2 lw $t1 0($t1) sw $t1 0($sp) addi $sp $sp -4	li $t1 0 sw $t1 0($sp) addi $sp $sp -4	lw $t1 4($sp) addi $sp $sp 4	lw $t2 4($sp) addi $sp $sp 4	sge $t1 $t2 $t1 sw $t1 0($sp) addi $sp $sp -4	lw $t1 4($sp) addi $sp $sp 4

src1.s (proj4/docs/src1_commented.s)



lw $ra 0($fp) **move $sp $fp** **lw $fp 4($sp)** **addi $sp $sp 4** **addi $sp $sp 4**

System Call

- SPIM provides a small set of OS-like services through the system call (syscall) instruction
  - To request a service, a program loads the system call code into register $v0 and arguments into registers $a0 - $a3
  - System calls that return values put their results in register $v0
- Example: print “the answer = 5”

```
    .data
str:
    .asciiz "the answer = ”
    .text
li $v0, 4           # system call code for print_str
la $a0, str         # address of string to print
syscall             # print the string

li $v0, 1           # system call code for print_int
li $a0, 5           # integer to print
syscall             # print it
```