

Brief Introduction of RISC-V ISA

Spring 2020

Outline

- Homework Overview
- Review Concept of ISA
- RISC-V ISA for Homework3

Homework Overview

- Implementing subset of C grammar for RISC-V platform
- Grading: 30%
 - HW1 – scanner (use lex tool)
 - 10%
 - HW2 – parser (use yacc + lex tool)
 - 10%
 - HW3 – code generator (yacc + lex + RISC-V)
 - 10% (Demo)

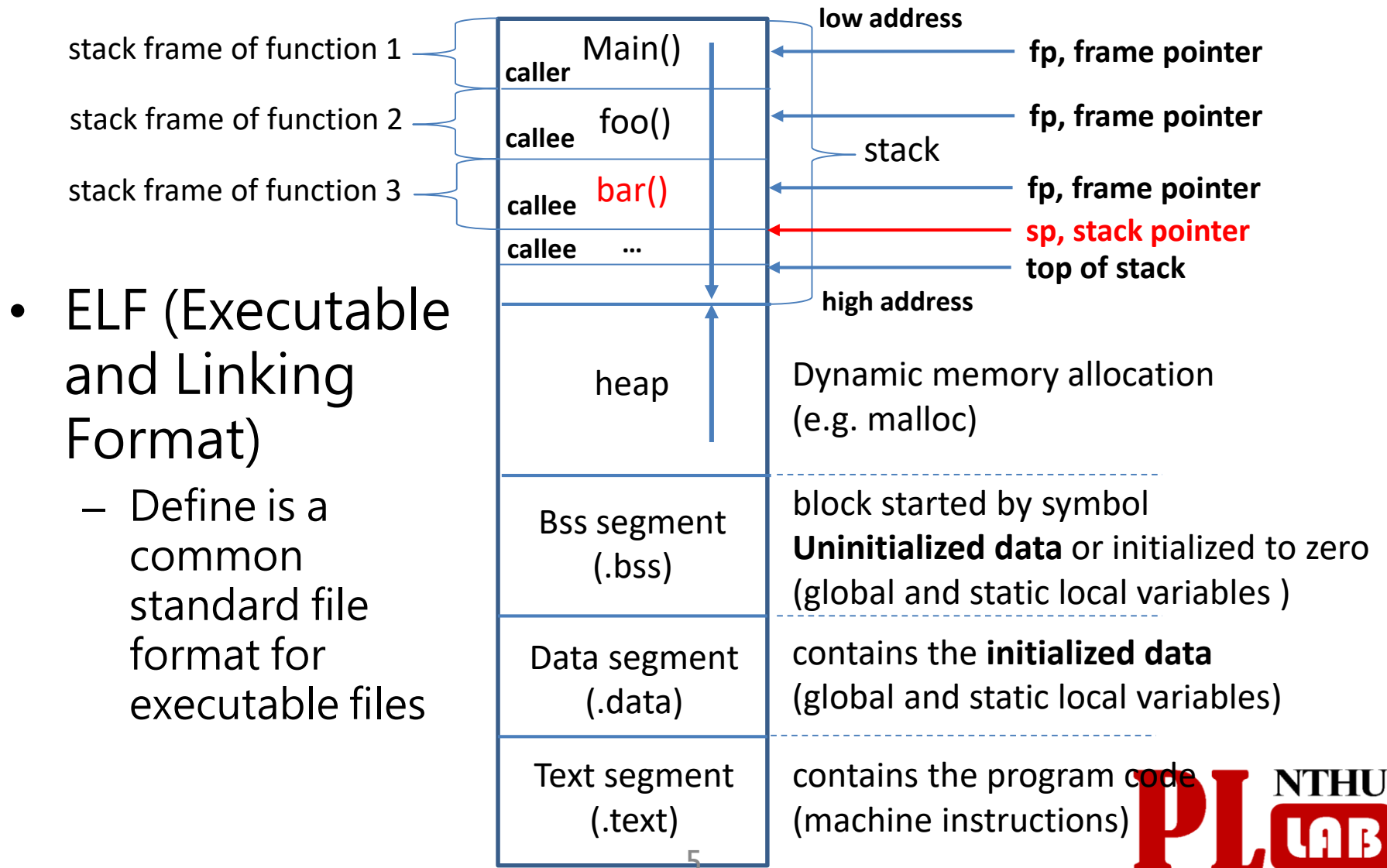
Review-

Basic Concept of ISA

- Computational Instructions
 - `op $rd, $rs, $rt`
 - `op $rd, $rs, constant` (Immediate mode)
- Data Transfer Instructions
 - `lw $rd, offset($rs)`
 - `sw $rs, offset($rd)`
- Program Control
 - `Jal label`
- `op`: opcode
- `$rd`: destination register
- `$rs`: 1st operand of source register
- `$rt`: 2nd operand of target register

Review-

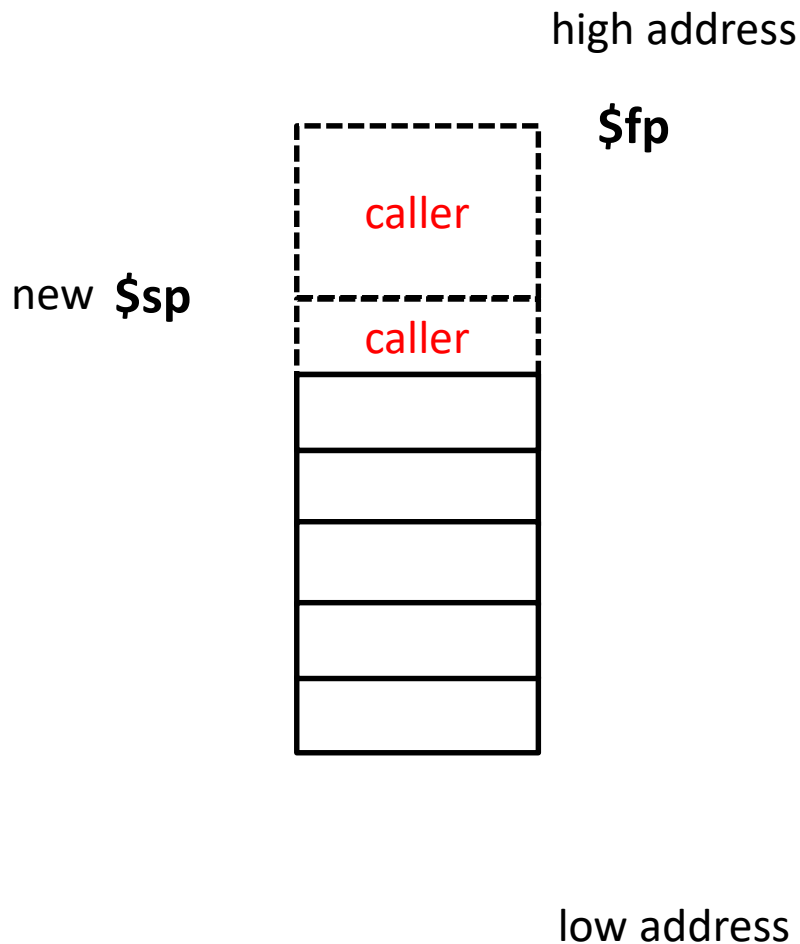
File Format and Call Stack



Review-Function Prologue

- Function starts working
 - create a stack frame for itself (change \$fp and \$sp)
 - save the return address in the stack frame
 - save any registers that it plans to change
- Changing \$fp and \$sp ...
 - new \$fp = old \$sp - 4
 - new \$sp = old \$sp - size of frame (in bytes)

Flow of Prologue



Foo ():

! Prologue

```
sw $fp, -4($sp)
```

```
sw $ra, -8($sp)
```

```
sw $s0, -12($sp)
```

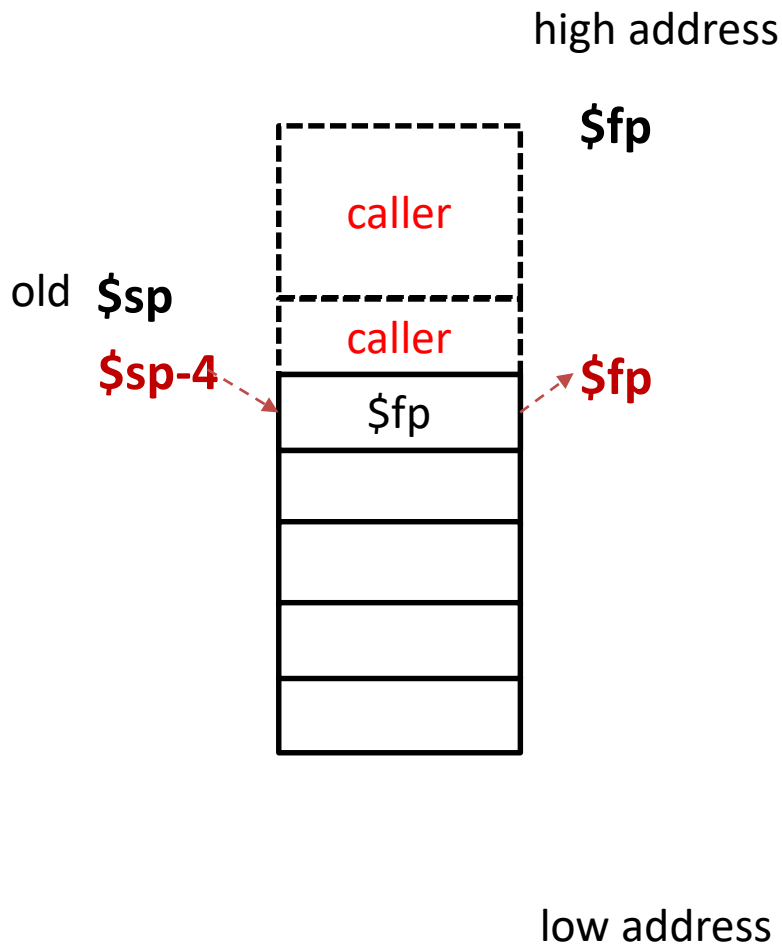
```
sw $s1, -16($sp)
```

```
sw $s2, -20($sp)
```

```
addi $sp, $sp, -20
```

! End of Prologue

Flow of Prologue



Foo ():

! Prologue

```
sw $fp, -4($sp)
```

```
sw $ra, -8($sp)
```

```
sw $s0, -12($sp)
```

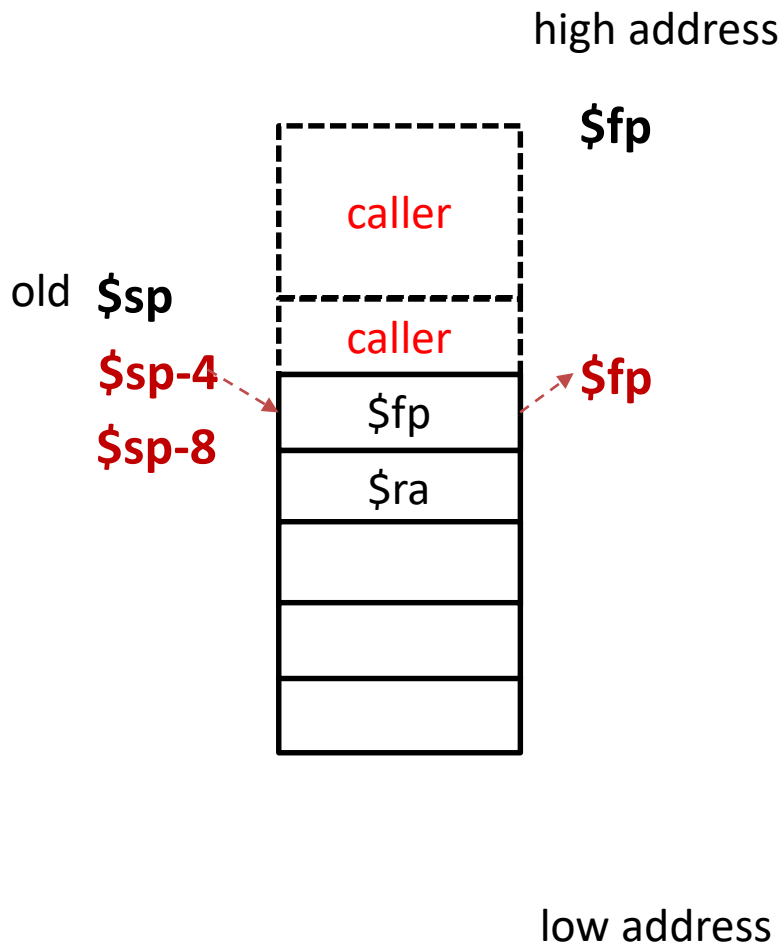
```
sw $s1, -16($sp)
```

```
sw $s2, -20($sp)
```

```
addi $sp, $sp, -20
```

! End of Prologue

Flow of Prologue



Foo ():

! Prologue

```
sw $fp, -4($sp)
```

```
sw $ra, -8($sp)
```

```
sw $s0, -12($sp)
```

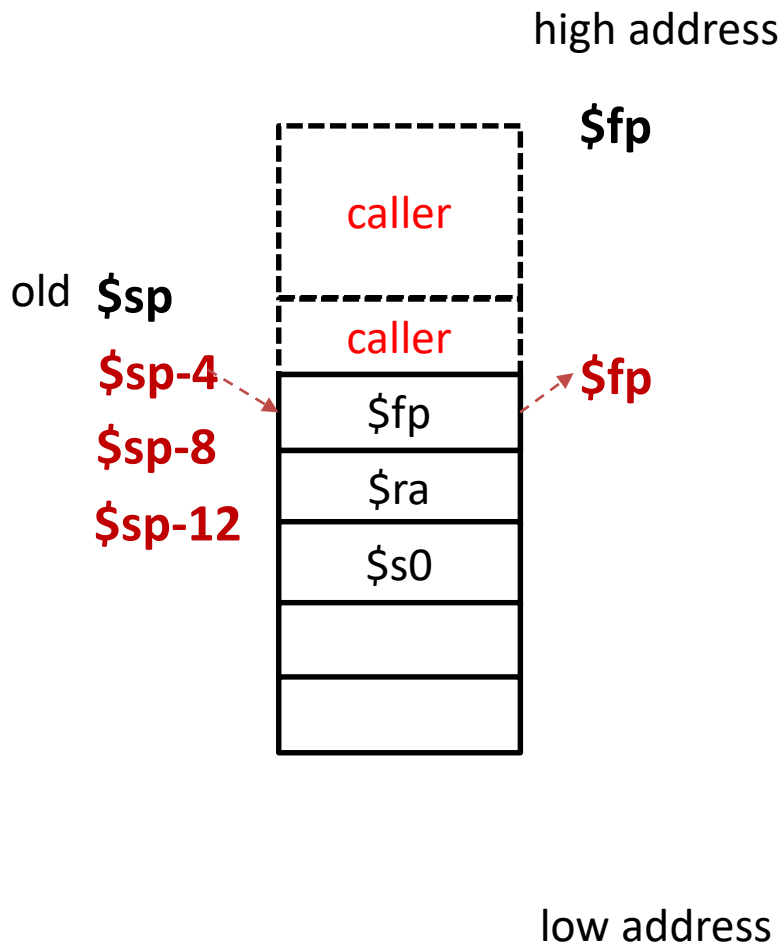
```
sw $s1, -16($sp)
```

```
sw $s2, -20($sp)
```

```
addi $sp, $sp, -20
```

! End of Prologue

Flow of Prologue



Foo ():

! Prologue

```
sw  $fp, -4($sp)
```

```
sw  $ra, -8($sp)
```

```
sw  $s0, -12($sp)
```

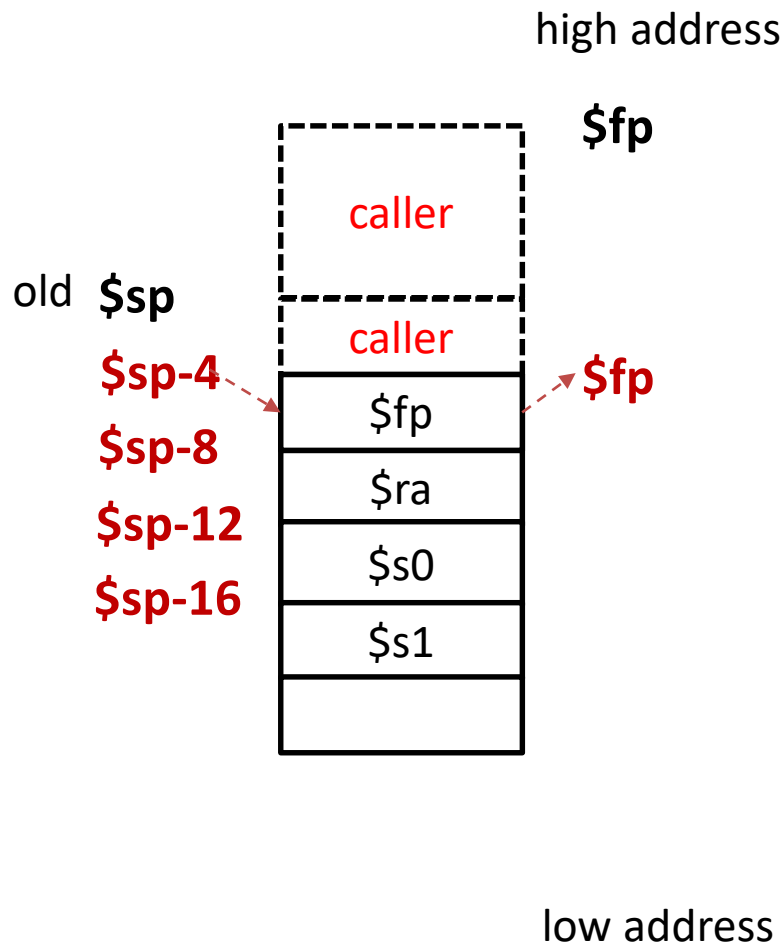
```
sw  $s1, -16($sp)
```

```
sw  $s2, -20($sp)
```

```
addi $sp, $sp, -20
```

! End of Prologue

Flow of Prologue



Foo ():

! Prologue

```
sw $fp, -4($sp)
```

```
sw $ra, -8($sp)
```

```
sw $s0, -12($sp)
```

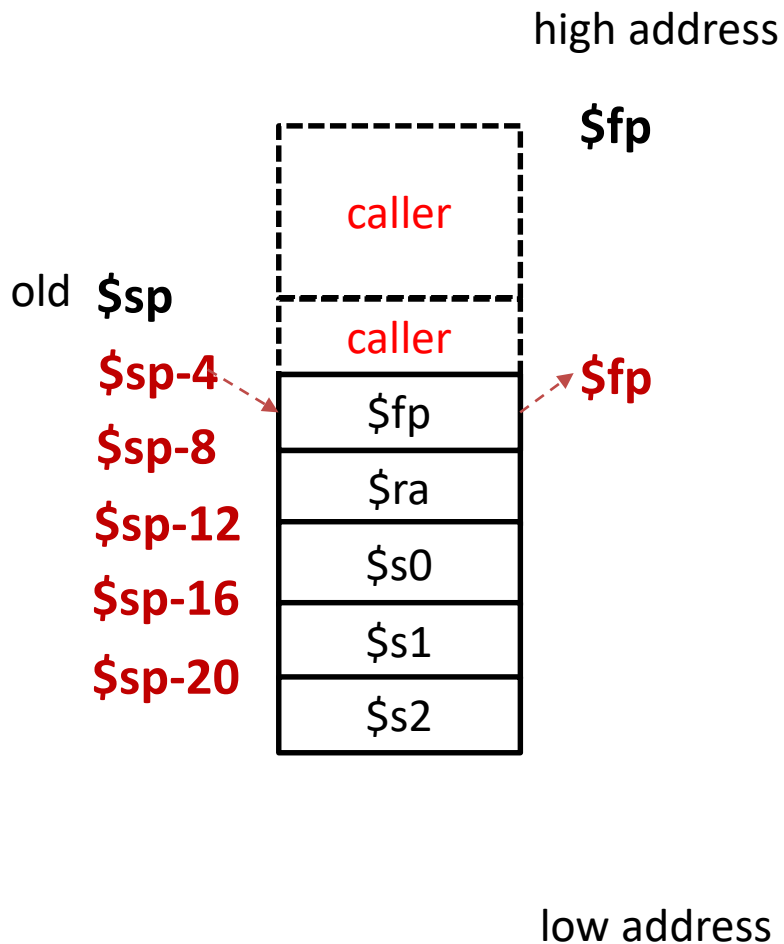
```
sw $s1, -16($sp)
```

```
sw $s2, -20($sp)
```

```
addi $sp, $sp, -20
```

! End of Prologue

Flow of Prologue



Foo ():

! Prologue

```
sw $fp, -4($sp)
```

```
sw $ra, -8($sp)
```

```
sw $s0, -12($sp)
```

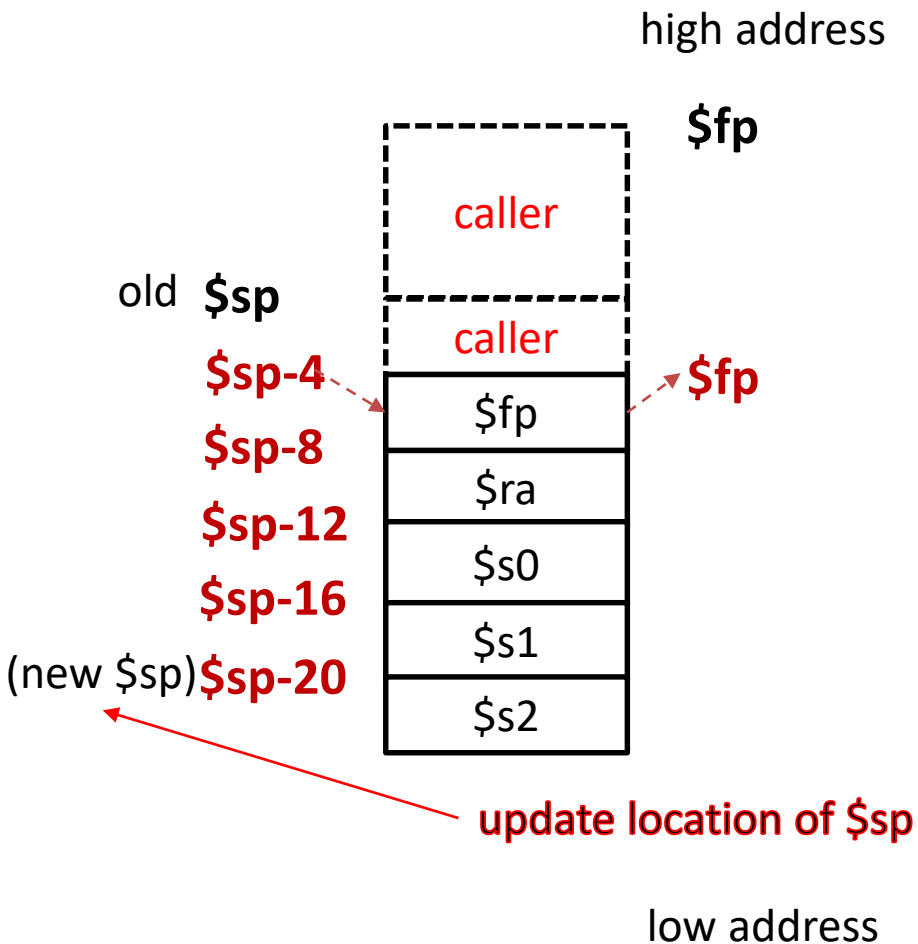
```
sw $s1, -16($sp)
```

```
sw $s2, -20($sp)
```

```
addi $sp, $sp, -20
```

! End of Prologue

Flow of Prologue



Foo ():

! Prologue

```
sw $fp, -4($sp)
```

```
sw $ra, -8($sp)
```

```
sw $s0, -12($sp)
```

```
sw $s1, -16($sp)
```

```
sw $s2, -20($sp)
```

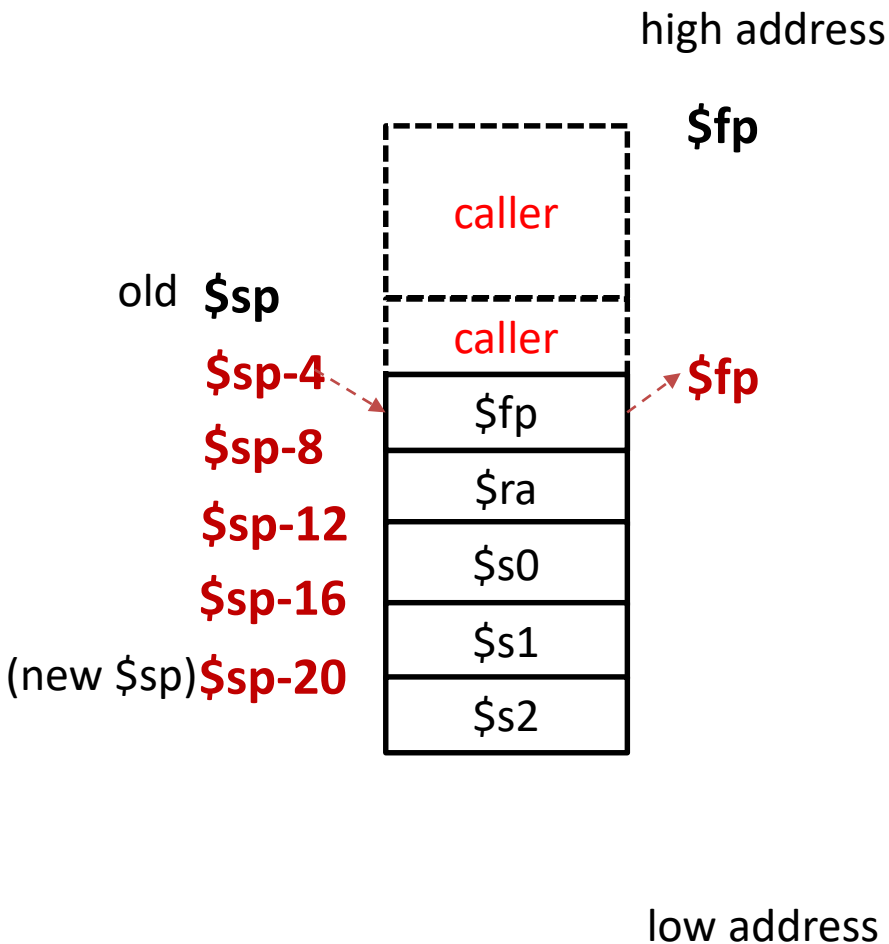
```
addi $sp, $sp, -20
```

! End of Prologue

Review-Function Epilogue

- Function end
 - pop any pushed arguments off the stack
 - restore the values of any saved registers
 - restore the saved value of \$ra (return address)
 - remove its stack frame (change \$fp and \$sp)
- Locations of saved values computed relative to \$fp
- Changing \$fp and \$sp ...
 - new \$sp = old \$fp + 4
 - new \$fp = memory[old \$fp]

Flow of Epilogue



Foo ():

! Epilogue

```
lw    $s2, -16($fp)
```

```
lw    $s1, -12($fp)
```

```
lw    $s0, -8($fp)
```

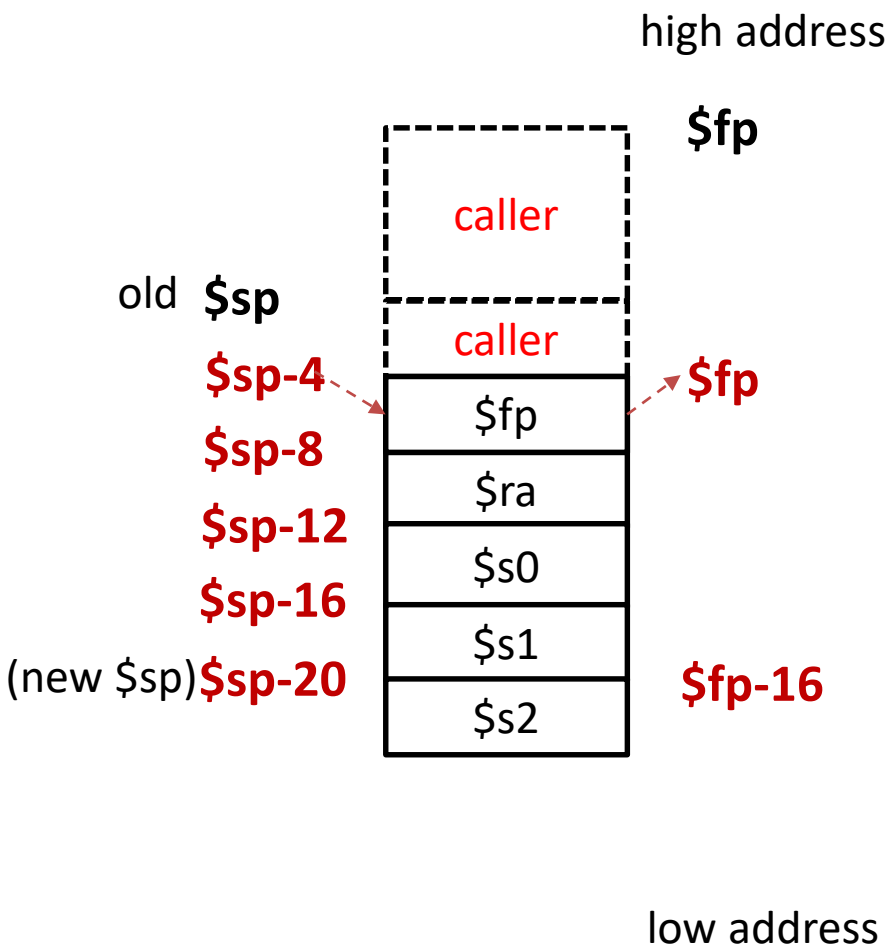
```
lw    $ra, -4($fp)
```

```
lw    $fp, ($fp)
```

```
addi $sp, $sp, 20
```

! End of Epilogue

Flow of Epilogue



Foo ():

! Epilogue

old sp



$\$fp = \$sp - 4$

lw \$s2, -16(\$fp)

lw \$s1, -12(\$fp)

lw \$s0, -8(\$fp)

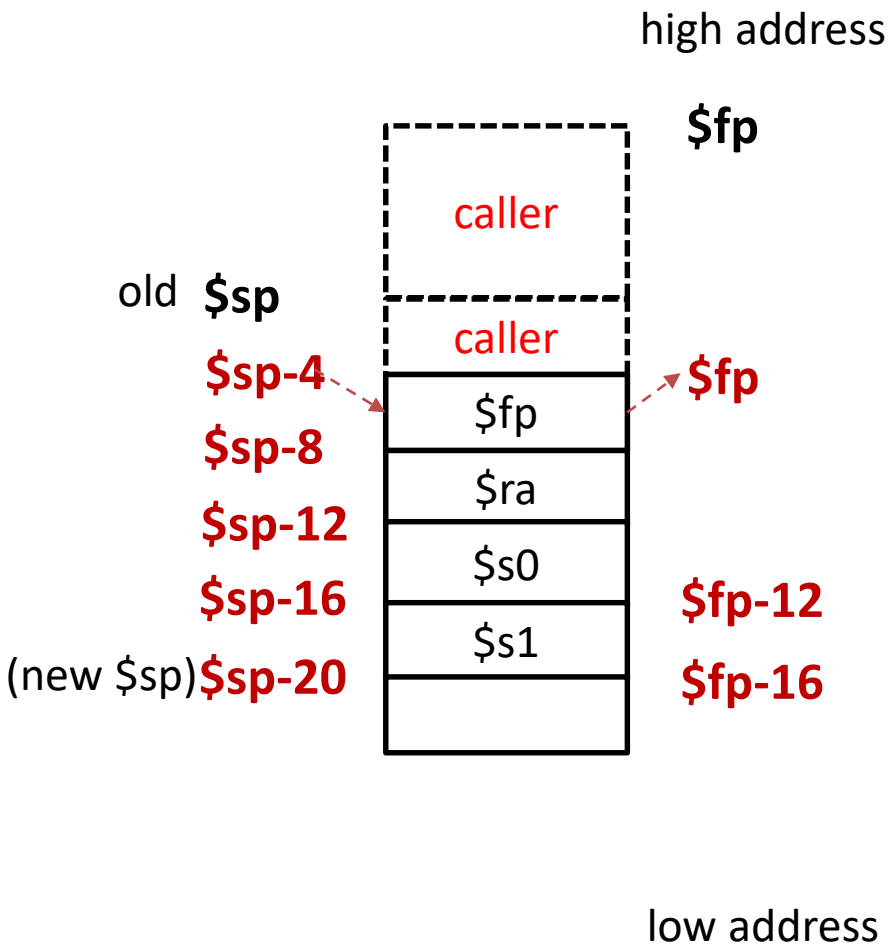
lw \$ra, -4(\$fp)

lw \$fp, (\$fp)

addi \$sp, \$sp, 20

! End of Epilogue

Flow of Epilogue



Foo ():

! Epilogue

$\$fp = \$sp - 4$

lw \$s2, -16(\$fp)

lw \$s1, -12(\$fp)

lw \$s0, -8(\$fp)

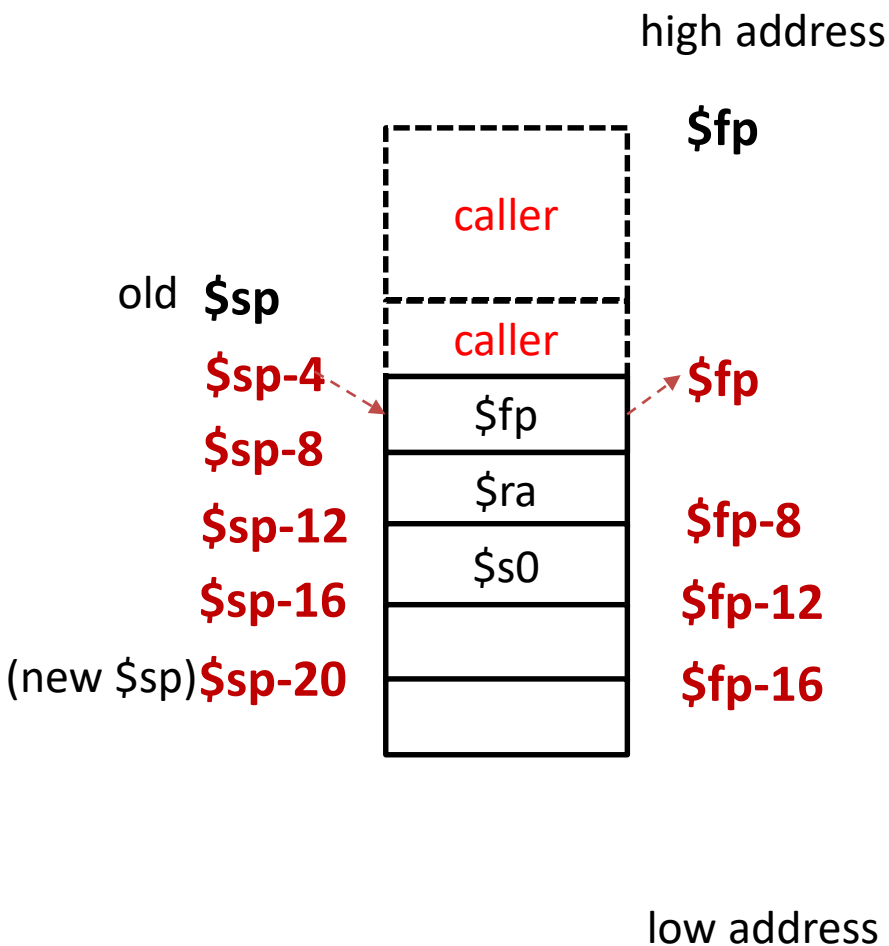
lw \$ra, -4(\$fp)

lw \$fp, (\$fp)

addi \$sp, \$sp, 20

! End of Epilogue

Flow of Epilogue



Foo ():

! Epilogue

$\$fp = \$sp-4$

lw \$s2, -16(\$fp)

lw \$s1, -12(\$fp)

lw \$s0, -8(\$fp)

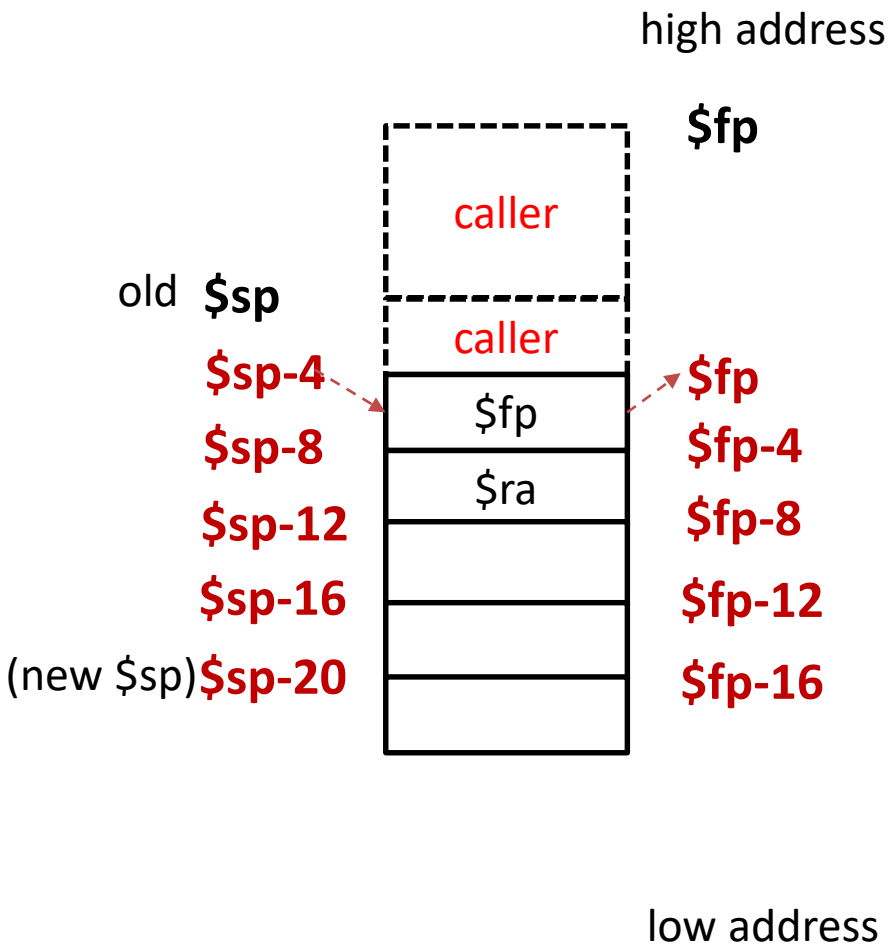
lw \$ra, -4(\$fp)

lw \$fp, (\$fp)

addi \$sp, \$sp, 20

! End of Epilogue

Flow of Epilogue



Foo ():

! Epilogue

$\$fp = \$sp-4$

lw \$s2, -16(\$fp)

lw \$s1, -12(\$fp)

lw \$s0, -8(\$fp)

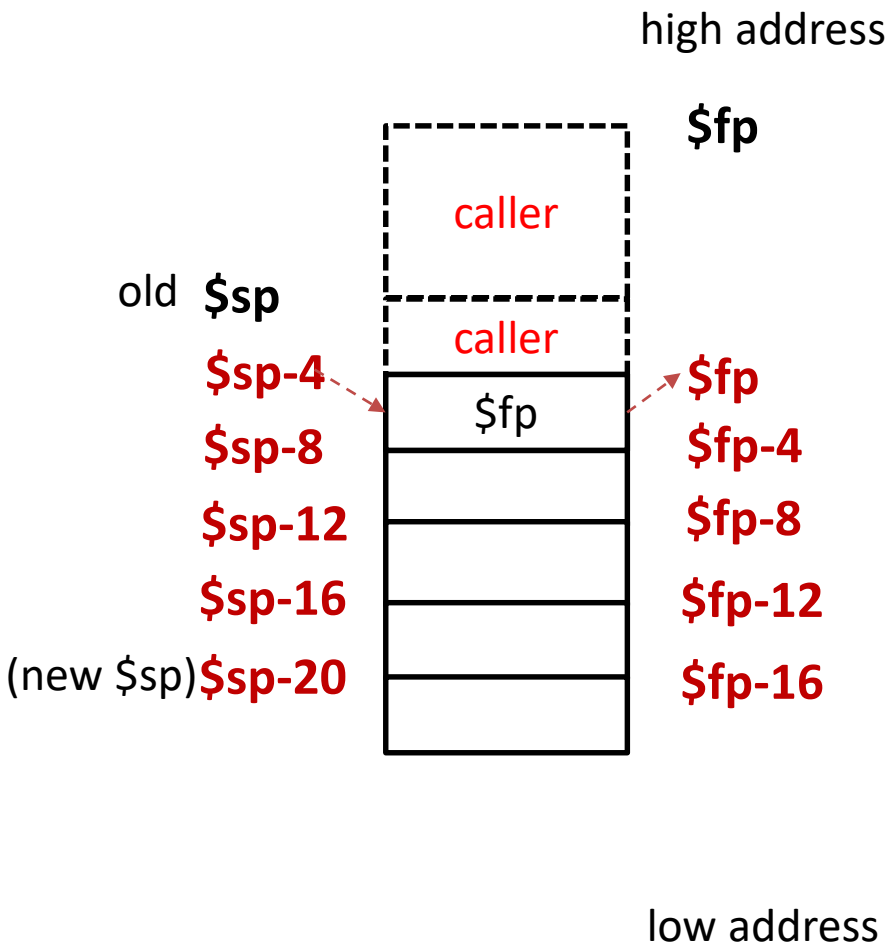
lw \$ra, -4(\$fp)

lw \$fp, (\$fp)

addi \$sp, \$sp, 20

! End of Epilogue

Flow of Epilogue



Foo ():

! Epilogue

$\$fp = \$sp-4$

lw \$s2, -16(\$fp)

lw \$s1, -12(\$fp)

lw \$s0, -8(\$fp)

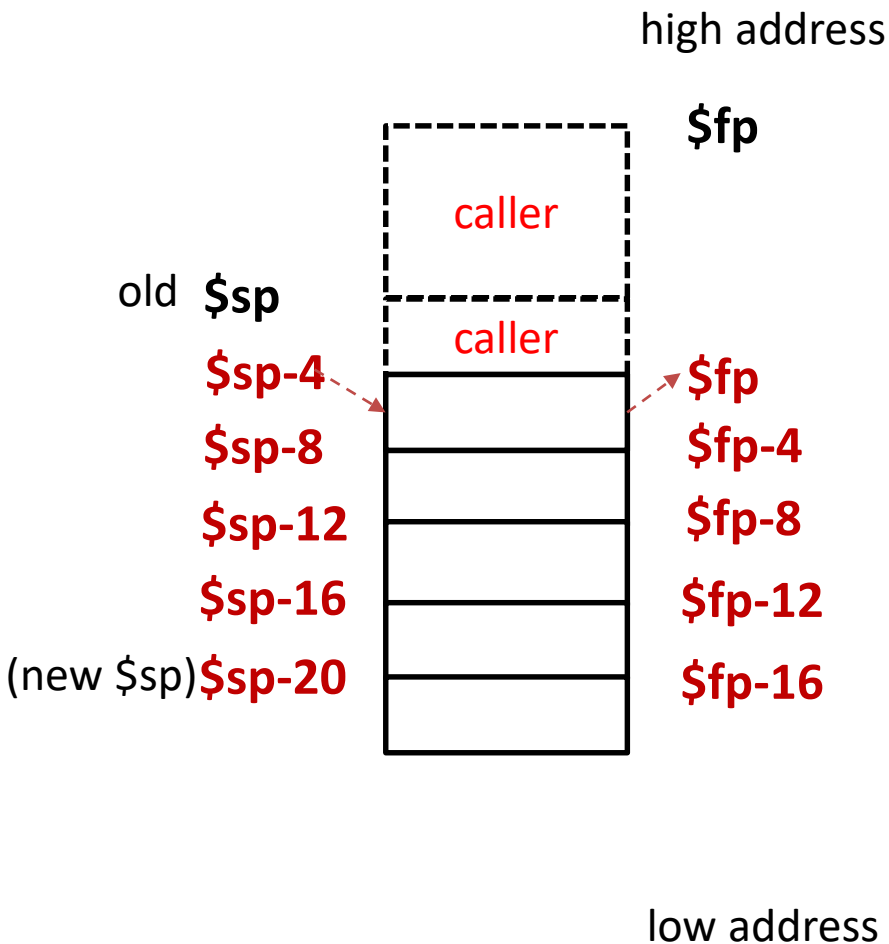
lw \$ra, -4(\$fp)

lw \$fp, (\$fp)

addi \$sp, \$sp, 20

! End of Epilogue

Flow of Epilogue



Foo ():

! Epilogue

$\$fp = \$sp-4$

lw $\$s2, -16(\$fp)$

lw $\$s1, -12(\$fp)$

lw $\$s0, -8(\$fp)$

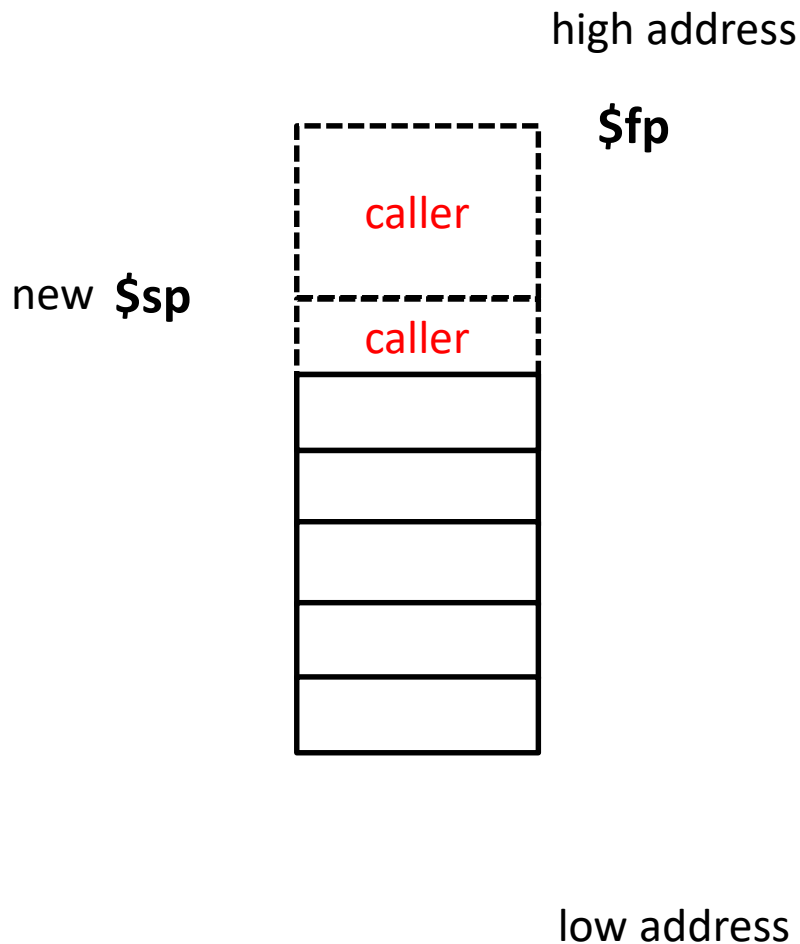
lw $\$ra, -4(\$fp)$

lw $\$fp, (\$fp)$

addi $\$sp, \$sp, 20$

! End of Epilogue

Flow of Epilogue



Foo ():

! Epilogue

$\$fp = \$sp - 4$

lw \$s2, -16(\$fp)

lw \$s1, -12(\$fp)

lw \$s0, -8(\$fp)

lw \$ra, -4(\$fp)

lw \$fp, (\$fp)

addi \$sp, \$sp, 20

! End of Epilogue

Introduction to RISC-V RV32I ISA Available for HW3

RISC-V

- RISC-V (pronounced "risk-five")
 - RISC-I, II, III, IV before
- Originated in 2010 by researchers at UC Berkeley
 - Krste Asanović, David Patterson and students
- Features
 - Full Free open-source
 - Allowing for user extensibility of the architecture
- Andes F1 board only support RV32I ISA of RISC-V

RISC-V General Purpose Registers

- 32 32-bit General Purpose Registers

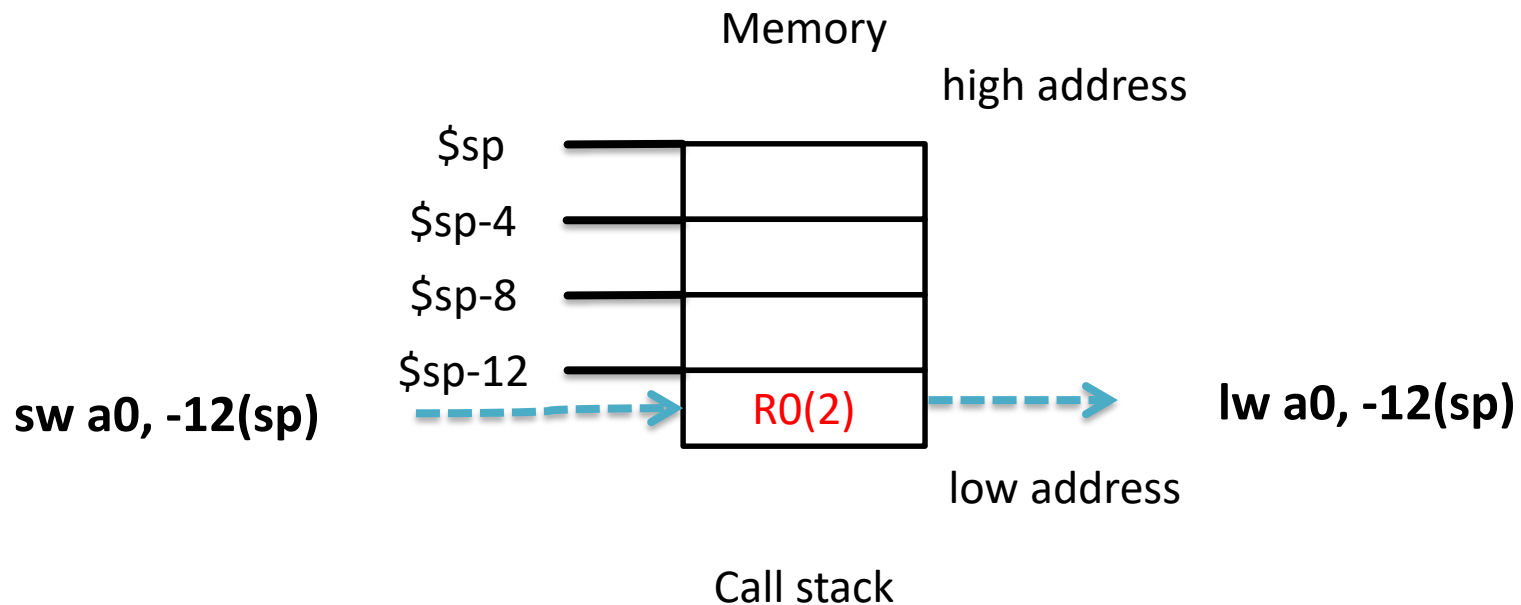
Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5–7	t0–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller
f0–7	ft0–7	FP temporaries	Caller
f8–9	fs0–1	FP saved registers	Callee
f10–11	fa0–1	FP arguments/return values	Caller
f12–17	fa2–7	FP arguments	Caller
f18–27	fs2–11	FP saved registers	Callee
f28–31	ft8–11	FP temporaries	Caller

Arithmetic ISA

指令	寫法	說明
add	add a0, a1, a2	$a0 = a1 + a2$
addi	addi a0, a1, 3	$a0 = a1 + 3$
sub	sub a0, a1, a2	$a0 = a1 - a2$
mul	mul a0, a1, a2	$a0 = a1 * a2$
div	div a0, a1, a2	$a0 = a1 / a2$
rem	rem a0, a1, a2	$a0 = a1 \% a2$

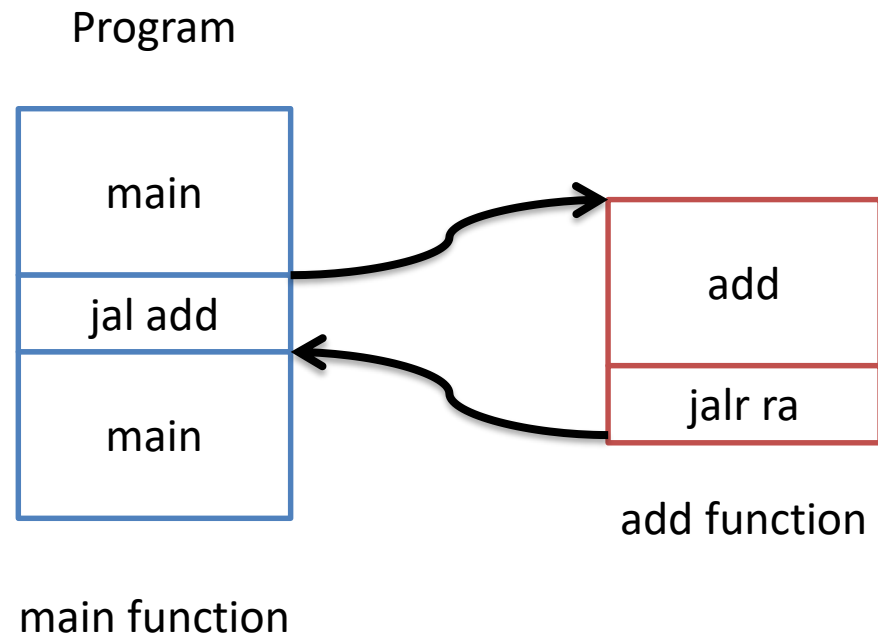
Load/Store ISA

指令	寫法	說明
li	li a0, 2	r0 = 2
lw	lw a0, -12(sp)	Load data from address sp-12 and store it into a0.
sw	sw a0, -12(sp)	Store a0 into address sp-12.



JMUP ISA

指令	寫法	說明
jal	jal add	Jump to 'add' function and link
jalr	jalr ra	Return from function.



Conditional Branch ISA

指令	寫法	說明
beq	beq a0, a1, offset	if a0 == a1 then jump
bne	bne a0, a1, offset	if a0 != a1 then jump
blt	blt a0, a1, offset	if a0 < a1 then jump
bge	bge a0, a1, offset	if a0 >= a1 then jump

Example of RISC-V Code

```

1 .global codegen
2 codegen:
3     // BEGIN PROLOGUE
4     // codegen is the callee here, so we save callee-saved registers
5     sw s0, -4(sp) // save frame pointer
6     addi sp, sp, -4
7     addi s0, sp, 0 // set new frame
8     sw sp, -4(s0)
9     sw s1, -8(s0)
10    sw s2, -12(s0)
11    sw s3, -16(s0)
12    sw s4, -20(s0)
13    sw s5, -24(s0)
14    sw s6, -28(s0)
15    sw s7, -32(s0)
16    sw s8, -36(s0)
17    sw s9, -40(s0)
18    sw s10, -44(s0)
19    sw s11, -48(s0)
20    addi sp, s0, -48 // update stack pointer
21    // END PROLOGUE

```

```

22
23    li t0, 1
24    sw t0, -4(sp)
25    addi sp, sp, -4
26    li t0, 2
27    sw t0, -4(sp)
28    addi sp, sp, -4
29    li t0, 1
30    sw t0, -4(sp)
31    addi sp, sp, -4
32    lw t0, 0(sp)
33    addi sp, sp, 4
34    lw t1, 0(sp)
35    addi sp, sp, 4
36    mul t0, t0, t1
37    sw t0, -4(sp)
38    addi sp, sp, -4
39    lw t0, 0(sp)
40    addi sp, sp, 4
41    lw t1, 0(sp)
42    addi sp, sp, 4
43    add t0, t0, t1
44    sw t0, -4(sp)
45    addi sp, sp, -4
46    lw t0, -52(s0)
47    sw t0, -4(sp)
48    addi sp, sp, -4
49    li t0, 3
50    sw t0, -4(sp)
51    addi sp, sp, -4
52    lw t0, 0(sp)
53    addi sp, sp, 4
54    lw t1, 0(sp)
55    addi sp, sp, 4
56    add t0, t0, t1
57    sw t0, -4(sp)
58    addi sp, sp, -4
59    li t0, 2
60    sw t0, -4(sp)
61    addi sp, sp, -4
62    lw t0, 0(sp)
63    addi sp, sp, 4
64    lw t1, 0(sp)
65    addi sp, sp, 4
66    div t0, t1, t0
67    sw t0, -4(sp)
68    addi sp, sp, -4

```

```

70    li t0, 1
71    sw t0, -4(sp)
72    addi sp, sp, -4
73    li t0, 26
74    sw t0, -4(sp)
75    addi sp, sp, -4
76    lw a0, 0(sp)
77    addi sp, sp, 4
78    lw a1, 0(sp)
79    addi sp, sp, 4
80    sw ra, -4(sp)
81    addi sp, sp, -4
82    jal ra, digitalWrite
83    lw ra, 0(sp)
84    addi sp, sp, 4
85
86    lw t0, -52(s0)
87    sw t0, -4(sp)
88    addi sp, sp, -4
89    li t0, 1000
90    sw t0, -4(sp)
91    addi sp, sp, -4
92    lw t0, 0(sp)
93    addi sp, sp, 4
94    lw t1, 0(sp)
95    addi sp, sp, 4
96    mul t0, t0, t1
97    sw t0, -4(sp)
98    addi sp, sp, -4
99    lw a0, 0(sp)
100    addi sp, sp, 4
101    sw ra, -4(sp)
102    addi sp, sp, -4
103    jal ra, delay
104    lw ra, 0(sp)
105    addi sp, sp, 4
106
107    li t0, 0
108    sw t0, -4(sp)
109    addi sp, sp, -4
110    li t0, 26
111    sw t0, -4(sp)
112    addi sp, sp, -4
113    lw a0, 0(sp)
114    addi sp, sp, 4
115    lw a1, 0(sp)
116    addi sp, sp, 4
117    sw ra, -4(sp)
118    addi sp, sp, -4
119    jal ra, digitalWrite
120    lw ra, 0(sp)
121    addi sp, sp, 4
122
123    lw t0, -56(s0)
124    sw t0, -4(sp)
125    addi sp, sp, -4
126    li t0, 1000
127    sw t0, -4(sp)
128    addi sp, sp, -4
129    lw t0, 0(sp)
130    addi sp, sp, 4
131    lw t1, 0(sp)
132    addi sp, sp, 4
133    mul t0, t0, t1
134    sw t0, -4(sp)
135    addi sp, sp, -4
136    lw a0, 0(sp)
137    addi sp, sp, 4
138    sw ra, -4(sp)
139    addi sp, sp, -4
140    jal ra, delay
141    lw ra, 0(sp)
142    addi sp, sp, 4
143
144

```

```

1 void codegen();
2 void codegen()
3 {
4     int a = 1 + 2 * 1; // a = 3
5     int b = (a + 3) / 2; // b = 3
6     digitalWrite(26, 1);
7     delay(a * 1000); // delay 3 seconds
8     digitalWrite(26, 0);
9     delay(b * 1000); // delay 3 seconds
10 }

```

```

145 // BEGIN EPILOGUE
146 // restore callee-saved registers
147 // s0 at this point should be the same as in prologue
148 lw s11, -48(s0)
149 lw s10, -44(s0)
150 lw s9, -40(s0)
151 lw s8, -36(s0)
152 lw s7, -32(s0)
153 lw s6, -28(s0)
154 lw s5, -24(s0)
155 lw s4, -20(s0)
156 lw s3, -16(s0)
157 lw s2, -12(s0)
158 lw s1, -8(s0)
159 lw sp, -4(s0)
160 addi sp, sp, 4
161 lw s0, -4(sp)
162 // END EPILOGUE
163
164 jalr zero, 0(ra) // return

```

Example of RISC-V Code

```
1  .global codegen
2  codegen:
3      // BEGIN PROLOGUE
4      // codegen is the callee here, so we save callee-saved registers
5      sw s0, -4(sp) // save frame pointer
6      addi sp, sp, -4
7      addi s0, sp, 0 // set new frame
8      sw sp, -4(s0)
9      sw s1, -8(s0)
10     sw s2, -12(s0)
11     sw s3, -16(s0)
12     sw s4, -20(s0)
13     sw s5, -24(s0)
14     sw s6, -28(s0)
15     sw s7, -32(s0)
16     sw s8, -36(s0)
17     sw s9, -40(s0)
18     sw s10, -44(s0)
19     sw s11, -48(s0)
20     addi sp, s0, -48 // update stack pointer
21     // END PROLOGUE
22
```

```
1  void codegen();
2  void codegen()
3  {
4      int a = 1 + 2 * 1; // a = 3
5      int b = (a + 3) / 2; // b = 3
6      digitalWrite(26, 1);
7      delay(a * 1000); // delay 3 seconds
8      digitalWrite(26, 0);
9      delay(b * 1000); // delay 3 seconds
10 }
```

Example of RISC-V Code

```
1  .global codegen
2  codegen:
3      // BEGIN PROLOGUE
4      // codegen is the callee here, so we save callee-saved registers
5      sw s0, -4(sp) // save frame pointer
6      addi sp, sp, -4
7      addi s0, sp, 0 // set new frame
8      sw sp, -4(s0)
9      sw s1, -8(s0)
10     sw s2, -12(s0)
11     sw s3, -16(s0)
12     sw s4, -20(s0)
13     sw s5, -24(s0)
14     sw s6, -28(s0)
15     sw s7, -32(s0)
16     sw s8, -36(s0)
17     sw s9, -40(s0)
18     sw s10, -44(s0)
19     sw s11, -48(s0)
20     addi sp, s0, -48 // update stack pointer
21     // END PROLOGUE
22
```

```
1  void codegen();
2  void codegen()
3  {
4      int a = 1 + 2 * 1; // a = 3
5      int b = (a + 3) / 2; // b = 3
6      digitalWrite(26, 1);
7      delay(a * 1000); // delay 3 seconds
8      digitalWrite(26, 0);
9      delay(b * 1000); // delay 3 seconds
10 }
```


Example of RISC-V Code

```
23 li t0, 1
24 sw t0, -4(sp)
25 addi sp, sp, -4
26 li t0, 2
27 sw t0, -4(sp)
28 addi sp, sp, -4
29 li t0, 1
30 sw t0, -4(sp)
31 addi sp, sp, -4
32 lw t0, 0(sp)
33 addi sp, sp, 4
34 lw t1, 0(sp)
35 addi sp, sp, 4
36 mul t0, t0, t1
37 sw t0, -4(sp)
38 addi sp, sp, -4
39 lw t0, 0(sp)
40 addi sp, sp, 4
41 lw t1, 0(sp)
42 addi sp, sp, 4
43 add t0, t0, t1
44 sw t0, -4(sp)
45 addi sp, sp, -4
46 lw t0, -52(s0)
47 sw t0, -4(sp)
48 addi sp, sp, -4
49 li t0, 3
50 sw t0, -4(sp)
51 addi sp, sp, -4
52 lw t0, 0(sp)
53 addi sp, sp, 4
54 lw t1, 0(sp)
55 addi sp, sp, 4
56 add t0, t0, t1
57 sw t0, -4(sp)
58 addi sp, sp, -4
59 li t0, 2
60 sw t0, -4(sp)
61 addi sp, sp, -4
62 lw t0, 0(sp)
63 addi sp, sp, 4
64 lw t1, 0(sp)
65 addi sp, sp, 4
66 div t0, t1, t0
67 sw t0, -4(sp)
68 addi sp, sp, -4
```

```
1 void codegen();
2 void codegen()
3 {
4   int a = 1 + 2 * 1; // a = 3
5   int b = (a + 3) / 2; // b = 3
6   digitalWrite(26, 1);
7   delay(a * 1000); // delay 3 seconds
8   digitalWrite(26, 0);
9   delay(b * 1000); // delay 3 seconds
10 }
```

Example of RISC-V Code

```
23 li t0, 1
24 sw t0, -4(sp)
25 addi sp, sp, -4
26 li t0, 2
27 sw t0, -4(sp)
28 addi sp, sp, -4
29 li t0, 1
30 sw t0, -4(sp)
31 addi sp, sp, -4
32 lw t0, 0(sp)
33 addi sp, sp, 4
34 lw t1, 0(sp)
35 addi sp, sp, 4
36 mul t0, t0, t1
37 sw t0, -4(sp)
38 addi sp, sp, -4
39 lw t0, 0(sp)
40 addi sp, sp, 4
41 lw t1, 0(sp)
42 addi sp, sp, 4
43 add t0, t0, t1
44 sw t0, -4(sp)
45 addi sp, sp, -4
46 lw t0, -52(s0)
47 sw t0, -4(sp)
48 addi sp, sp, -4
49 li t0, 3
50 sw t0, -4(sp)
51 addi sp, sp, -4
52 lw t0, 0(sp)
53 addi sp, sp, 4
54 lw t1, 0(sp)
55 addi sp, sp, 4
56 add t0, t0, t1
57 sw t0, -4(sp)
58 addi sp, sp, -4
59 li t0, 2
60 sw t0, -4(sp)
61 addi sp, sp, -4
62 lw t0, 0(sp)
63 addi sp, sp, 4
64 lw t1, 0(sp)
65 addi sp, sp, 4
66 div t0, t1, t0
67 sw t0, -4(sp)
68 addi sp, sp, -4
```

```
1 void codegen();
2 void codegen()
3 {
4     int a = 1 + 2 * 1; // a = 3
5     int b = (a + 3) / 2; // b = 3
6     digitalWrite(26, 1);
7     delay(a * 1000); // delay 3 seconds
8     digitalWrite(26, 0);
9     delay(b * 1000); // delay 3 seconds
10 }
```

Example of RISC-V Code

```
69  li t0, 1
70  sw t0, -4(sp)
71  addi sp, sp, -4
72  li t0, 26
73  sw t0, -4(sp)
74  addi sp, sp, -4
75  lw a0, 0(sp)
76  addi sp, sp, 4
77  lw a1, 0(sp)
78  addi sp, sp, 4
79  sw ra, -4(sp)
80  addi sp, sp, -4
81  jal ra, digitalWrite
82  lw ra, 0(sp)
83  addi sp, sp, 4
```

```
85  lw t0, -52(sp)
86  sw t0, -4(sp)
87  addi sp, sp, -4
88  li t0, 1000
89  sw t0, -4(sp)
90  addi sp, sp, -4
91  lw t0, 0(sp)
92  addi sp, sp, 4
93  lw t1, 0(sp)
94  addi sp, sp, 4
95  mul t0, t0, t1
96  sw t0, -4(sp)
97  addi sp, sp, -4
98  lw a0, 0(sp)
99  addi sp, sp, 4
100 sw ra, -4(sp)
101 addi sp, sp, -4
102 jal ra, delay
103 lw ra, 0(sp)
104 addi sp, sp, 4
105
106
```

```
1  void codegen();
2  void codegen()
3  {
4      int a = 1 + 2 * 1; // a = 3
5      int b = (a + 3) / 2; // b = 3
6      digitalWrite(26, 1);
7      delay(a * 1000); // delay 3 seconds
8      digitalWrite(26, 0);
9      delay(b * 1000); // delay 3 seconds
10 }
```

Example of RISC-V Code

```
69  li t0, 1
70  sw t0, -4(sp)
71  addi sp, sp, -4
72  li t0, 26
73  sw t0, -4(sp)
74  addi sp, sp, -4
75  lw a0, 0(sp)
76  addi sp, sp, 4
77  lw a1, 0(sp)
78  addi sp, sp, 4
79  sw ra, -4(sp)
80  addi sp, sp, -4
81  jal ra, digitalWrite
82  lw ra, 0(sp)
83  addi sp, sp, 4
```

```
85  lw t0, -52(sp)
86  sw t0, -4(sp)
87  addi sp, sp, -4
88  li t0, 1000
89  sw t0, -4(sp)
90  addi sp, sp, -4
91  lw t0, 0(sp)
92  addi sp, sp, 4
93  lw t1, 0(sp)
94  addi sp, sp, 4
95  mul t0, t0, t1
96  sw t0, -4(sp)
97  addi sp, sp, -4
98  lw a0, 0(sp)
99  addi sp, sp, 4
100 sw ra, -4(sp)
101 addi sp, sp, -4
102 jal ra, delay
103 lw ra, 0(sp)
104 addi sp, sp, 4
105
106
```

```
1  void codegen();
2  void codegen()
3  {
4      int a = 1 + 2 * 1; // a = 3
5      int b = (a + 3) / 2; // b = 3
6      digitalWrite(26, 1);
7      delay(a * 1000); // delay 3 seconds
8      digitalWrite(26, 0);
9      delay(b * 1000); // delay 3 seconds
10 }
```

Example of RISC-V Code

```
107 li t0, 0
108 sw t0, -4(sp)
109 addi sp, sp, -4
110 li t0, 26
111 sw t0, -4(sp)
112 addi sp, sp, -4
113 lw a0, 0(sp)
114 addi sp, sp, 4
115 lw a1, 0(sp)
116 addi sp, sp, 4
117 sw ra, -4(sp)
118 addi sp, sp, -4
119 jal ra, digitalWrite
120 lw ra, 0(sp)
121 addi sp, sp, 4
122
123 lw t0, -56(s0)
124 sw t0, -4(sp)
125 addi sp, sp, -4
126 li t0, 1000
127 sw t0, -4(sp)
128 addi sp, sp, -4
129 lw t0, 0(sp)
130 addi sp, sp, 4
131 lw t1, 0(sp)
132 addi sp, sp, 4
133 mul t0, t0, t1
134 sw t0, -4(sp)
135 addi sp, sp, -4
136 lw a0, 0(sp)
137 addi sp, sp, 4
138 sw ra, -4(sp)
139 addi sp, sp, -4
140 jal ra, delay
141 lw ra, 0(sp)
142 addi sp, sp, 4
143
144
```

```
1 void codegen();
2 void codegen()
3 {
4     int a = 1 + 2 * 1; // a = 3
5     int b = (a + 3) / 2; // b = 3
6     digitalWrite(26, 1);
7     delay(a * 1000); // delay 3 seconds
8     digitalWrite(26, 0);
9     delay(b * 1000); // delay 3 seconds
10 }
```

Example of RISC-V Code

```
107 li t0, 0
108 sw t0, -4(sp)
109 addi sp, sp, -4
110 li t0, 26
111 sw t0, -4(sp)
112 addi sp, sp, -4
113 lw a0, 0(sp)
114 addi sp, sp, 4
115 lw a1, 0(sp)
116 addi sp, sp, 4
117 sw ra, -4(sp)
118 addi sp, sp, -4
119 jal ra, digitalWrite
120 lw ra, 0(sp)
121 addi sp, sp, 4
122
123 lw t0, -56(s0)
124 sw t0, -4(sp)
125 addi sp, sp, -4
126 li t0, 1000
127 sw t0, -4(sp)
128 addi sp, sp, -4
129 lw t0, 0(sp)
130 addi sp, sp, 4
131 lw t1, 0(sp)
132 addi sp, sp, 4
133 mul t0, t0, t1
134 sw t0, -4(sp)
135 addi sp, sp, -4
136 lw a0, 0(sp)
137 addi sp, sp, 4
138 sw ra, -4(sp)
139 addi sp, sp, -4
140 jal ra, delay
141 lw ra, 0(sp)
142 addi sp, sp, 4
143
144
```

```
1 void codegen();
2 void codegen()
3 {
4     int a = 1 + 2 * 1; // a = 3
5     int b = (a + 3) / 2; // b = 3
6     digitalWrite(26, 1);
7     delay(a * 1000); // delay 3 seconds
8     digitalWrite(26, 0);
9     delay(b * 1000); // delay 3 seconds
10 }
```

Example of RISC-V Code

```
145 // BEGIN EPILOGUE
146 // restore callee-saved registers
147 // s0 at this point should be the same as in prologue
148 lw s11, -48(s0)
149 lw s10, -44(s0)
150 lw s9, -40(s0)
151 lw s8, -36(s0)
152 lw s7, -32(s0)
153 lw s6, -28(s0)
154 lw s5, -24(s0)
155 lw s4, -20(s0)
156 lw s3, -16(s0)
157 lw s2, -12(s0)
158 lw s1, -8(s0)
159 lw sp, -4(s0)
160 addi sp, sp, 4
161 lw s0, -4(sp)
162 // END EPILOGUE
163
164 jalr zero, 0(ra) // return
```

```
1 void codegen();
2 void codegen()
3 {
4     int a = 1 + 2 * 1; // a = 3
5     int b = (a + 3) / 2; // b = 3
6     digitalWrite(26, 1);
7     delay(a * 1000); // delay 3 seconds
8     digitalWrite(26, 0);
9     delay(b * 1000); // delay 3 seconds
10 }
```

End