Computer Structure and Language

Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran

1

RISC-V Documents

- Programmers handbook included in RISC-V specification
  - https://riscv.org/technical/specifications/
- RISC-V Calling Convention
  - https://riscv.org/wp-content/uploads/2015/01/riscv-calling.pdf
  - https://wiki.riscv.org/display/HOME/RISC-V+Technical+Specifications
  - https://drive.google.com/file/d/1Ja_Tpp_5Me583CGVD-BIZMIgGBnlKU4R/view

2

RISC-V Conventional Register Usage

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

3

RISC-V Calling Convention

- Input, output, callee and caller saved registers are as specified before
- Arguments that are larger than one register and fit in two registers are passed in even-odd pair registers (like a2,a3)
- If arguments don't fit into registers, they are passed on stack
- Stack grows downwards
- Stack is always aligned by 16 bytes
- If return value is larger than one register and fits in two registers, it is return in two specified registers
- In case a value takes two registers, the lower part is put into lower numbered register and higher part is put into higher numbered register
- Consult documentation for other details

4

SAMPLE CODES

5

String Length – C Function

```c
int strlen(const char *str) {
    int i;
    for (i = 0; str[i] != '\0'; i++);
    return i;
}
```

6

String Length – RISC-V Function

```
.section .text
.global strlen
strlen:
    # a0 = const char *str
    li      t0, 0           # i = 0
loop_start: # Start of for loop
    add     t1, t0, a0      # t1 = i + str = &str[i]
    lb      t1, 0(t1)       # t1 = str[i]
    beqz    t1, loop_end    # if str[i] == 0, break for loop
    addi    t0, t0, 1       # Add 1 to iterator
    j       loop_start      # Jump back to condition
loop_end: # End of for loop
    mv      a0, t0          # Move t0 into a0 to return
    ret                     # Return back via the return address register
```

7

String Length – RISC-V Function Another Label Syntax

```
.section .text
.global strlen
strlen:
    # a0 = const char *str
    li      t0, 0           # i = 0
1: # Start of for loop
    add     t1, t0, a0      # t1 = i + str = &str[i]
    lb      t1, 0(t1)       # t1 = str[i]
    beqz    t1, 1f          # if str[i] == 0, break for loop
    addi    t0, t0, 1       # Add 1 to iterator
    j       1b              # Jump back to condition
1: # End of for loop
    mv      a0, t0          # Move t0 into a0 to return
    ret                     # Return back via the return address register
```

8

String Copy – C Function

```c
void stringcopy(char *dst, const char *src) {
    do {
      *dst = *src;
      if (*src == '\0') break;
      dst++;
      src++;
    } while (true);
}
```

9

String Copy – RISC-V Function

```
.section .text
.global stringcopy
stringcopy:
    # a0 = dst
    # a1 = src
loop_start:
    lb      t0, 0(a1)    # t0 = *src
    sb      t0, 0(a0)    # *dst = t0
    beqz    t0, loop_end # break if t0 == 0

    addi    a0, a0, 1    # a0(dst) += 1
    addi    a1, a1, 1    # a1(src) += 1
    j       loop_start
loop_end:
    ret
```

10

Sum of Integer Array – C Function

```c
int arraysum(int a[], int size) {
    int ret = 0;
    int i;
    for (i = 0;i < size;i++) {
        ret = ret + a[i];
    }
    return ret;
}
```

11

Sum of Integer Array – RISC-V Function

```asm
.section .text
.global arraysum
arraysum:
    # a0 = int a[], a1 = int size
    # t0 = sum, t1 = i
    li    t0, 0            # sum = 0
    li    t1, 0            # i = 0
loop_start:  # For loop
    bge   t1, a1, loop_end # if i >= size, break
    slli  t2, t1, 2        # Multiply i by 4 (1 << 2 = 4)
    add   t2, a0, t2       # calculate &a[i] = a + i * 4
    lw    t2, 0(t2)        # t2 = a[i]
    add   t0, t0, t2       # t0(sum) += t2(a[i])
    addi  t1, t1, 1        # i++
    j     loop_start
loop_end:
    mv    a0, t0       # a0(return value) = t0(sum)
    ret
```

12

Bubble Sort – RISC-V Function

```
.section .text
.global bubblesort
bubblesort:
    # a0 = a[] , a1 = len , t0 = changed , t1 = i
li t0, 1                        # changed = true
outer_loop_start:
    beq a1, zero, outer_loop_end   # if len == 0, break
    beq t0, zero, outer_loop_end   # if !changed, break
    xor t0, t0, t0                 # changed = false
    li t1, 1                       # i = 1
    inner_loop_start:
        beq t1, a1, inner_loop_end # if i == size, break
        slli t2, t1, 2             # t2 = i << 2
        add t2, t2, a0             # t2 = a + i * 4 = &a[i]
        lw t3, 0(t2)               # t3 = a[i]
        lw t4, -4(t2)              # t4 = a[i - 1]
        bge t3, t4, skip_swap
            sw t3, -4(t2)          # a[i - 1] = t3
            sw t4, 0(t2)           # a[i] = t4
            li t0, 1               # changed = true
        skip_swap:
        addi t1, t1, 1             # t1 (i) += 1
        j inner_loop_start
    inner_loop_end:
    addi a1, a1, -1                # len -= 1
    j outer_loop_start
outer_loop_end:
    ret
```

13

Float Distance – C Function

```
float distance(float x1, float y1, float x2, float y2) {
    float x_dist = x1 - x2;
    float y_dist = y1 - y2;
    return sqrt(x_dist * x_dist + y_dist * y_dist);
}
```

14

Float Distance – RISC-V Function

```
.section .text
.global distance
distance:
    # fa0 = x1 , fa1 = y1 , fa2 = x2 , fa3 = y2

    fsub.s ft0, fa0, fa2 # ft0 = fa0 - fa2 (x1 - x2)
    fsub.s ft1, fa1, fa3 # ft1 = fa1 - fa3 (y1 - y2)
    fmul.s ft0, ft0, ft0 # ft0 = ft0 * ft0
    fmul.s ft1, ft1, ft1 # ft0 = ft0 * ft0
    fadd.s ft2, ft0, ft1 # ft2 = ft1 + ft2 = dist ** 2
    fsqrt.s fa0, ft2     # fa0 = sqrt(ft2) = dist

    ret
```

15

Calling Function – RISC-V Function (P.1)

```
.section .text
.global func

func:
    # return a0 * a1 + a2
    mul a0, a0, a1
    add a0, a0, a2
    ret
```

16

Calling Function – RISC-V Function (P.2)

```
.section .rodata
scan: .asciz "%d %d %d"
result_out: .asciz "Result = %d\n"

.section .text
.global main
main:
    addi    sp, sp, -16     # Allocate 16 bytes from the stack
    sw      ra, 0(sp)       # Saving current ra
    # Calling scanf
    la      a0, scan        # Loading format
    addi    a1, sp, 4       # Address of a is sp + 4
    addi    a2, sp, 8       # Address of b is sp + 8
    addi    a3, sp, 12      # Address of c is sp + 12
    call    scanf
    # Loading values from memory to registers and calling func
    lw      a0, 4(sp)
    lw      a1, 8(sp)
    lw      a2, 12(sp)
    call    func
    # The result should be in a0, but that needs to be
    # the second parameter to printf.
    mv      a1, a0
    la      a0, result_out
    call    printf
    # Restore original RA and return
    lw      ra, 0(sp)
    addi    sp, sp, 16      # deallocate the stack
    ret
```

17

END OF SLIDES

18