Discussion 4 RISC-V

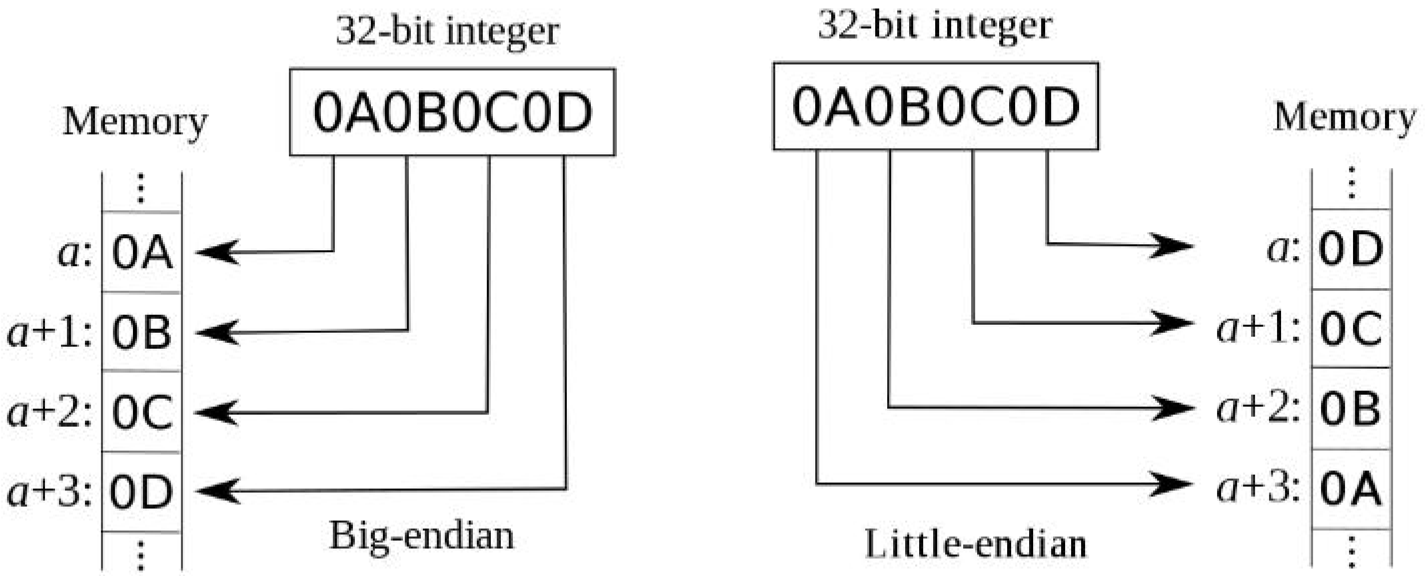
Agenda

1. Big Endian vs. Little Endian
2. Label and Assembler Directives
3. Enviroment calls
4. RISC-V Practices
5. Venus Q&A

Big Endian vs. Little Endian

MSB : the highest-order bit in a binary number. It represents the most significant part of the value.
LSB : the lowest-order bit in a binary number. It represents the least significant part of the value.

Big-Endian: MSB at lowest address
Little-Endian: LSB at lowest address



Example:
0x0A0B0C0D
MSB = 0x0A
LSB = 0x0D

Label and Assembler Directives

Directives are instructions to the assembler that tell it how to organize code, data, and symbols.
Label:a label is a symbolic name for the address of a data item or an instruction.

`.text`: Subsequent items put in user text segment (machine code)

`.data`: Subsequent items put in user data segment (binary rep of data in source file)

`.globl sym`: declares `sym` global and can be referenced from other files

`.asciiz str`: Store the string `str` in memory and null-terminates it

`.word` $w_1 \ldots w_n$: Store the *n* 32-bit quantities in successive memory words

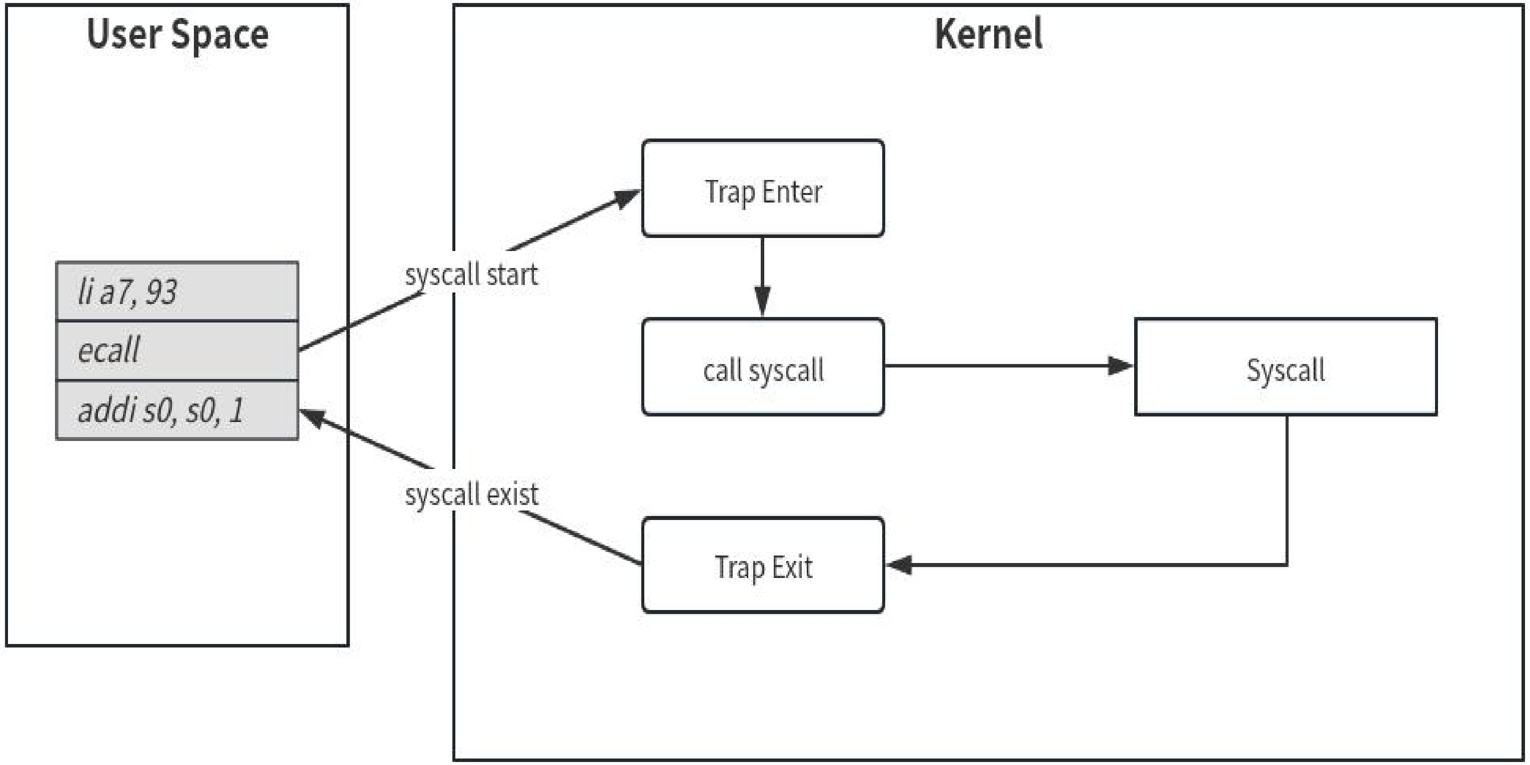
Label and Assembler Directives

```
.data
course:
    .asciiz "cs110"

semester:
    .asciiz "sp26"

num:
    .word 2026

.text
    la a1, course
    addi a0, x0, 4      # ecall 4 -- print_string
    ecall

    addi a1, x0, 10     # ASCII 10 -- '\n'
    addi a0, x0, 11     # ecall 11 -- print_character
    ecall

    la a1, semester
    addi a0, x0, 4      # ecall 4 -- print_string
    ecall

    addi a0, x0, 10     # ecall 10 -- exit
    ecall
```

label : course , semester,num

assembler directives
- section management : data,text
- data definition : asciiz , word

Ecall



Dual mode
- User program runs in user model
- OS services run in kernel mode

Triggers a trap
- Switches to kernel mode
- Kernel handles the request
- Returns to user program

```
.data
course:
    .asciiz "cs110"

semester:
    .asciiz "sp26"

num:
    .word 2026

.text
    la a1, course
    addi a0, x0, 4      # ecall 4 -- print_string
    ecall

    addi a1, x0, 10     # ASCII 10 -- '\n'
    addi a0, x0, 11     # ecall 11 -- print_character
    ecall

    la a1, semester
    addi a0, x0, 4      # ecall 4 -- print_string
    ecall

    addi a0, x0, 10     # ecall 10 -- exit
    ecall
```

How to Use ecall

- a0 → syscall number
- a1 → argument

Output

```
cs110
sp26
```

RISC-V Practices

4. **RISC-V**

8

(a) Consider the following code snippet written in RISC-V. The function `Factorial` is to calculate the factorial of a given number. (i.e. $n! = n \cdot (n-1) \cdots 2 \cdot 1$)

```
1  Factorial:
2      addi    sp, sp, -8
3      sw      ra, 0(sp)
4      li      t0, 1
5      beq     a0, t0, last_sit
6      sw      a0, 4(sp)
7      ________________
8      ________________
9      lw      t0, 4(sp)
10     ________________
11     j       fact_done
12 last_sit:
13     ________________
14 fact_done:
15     lw      ra, 0(sp)
16     addi    sp, sp, 8
17     mv      a0, a1
18     jr      ra
```

Fill in the missing code below.

line 7: ______________________

line 8: jal Factorial

line 10: ______________________

line 13: ______________________

RISC-V Practices

上海科技大学 ShanghaiTech University

4. **RISC-V**

[8] (a) Consider the following code snippet written in RISC-V. The function `Factorial` is to calculate the factorial of a given number. (i.e. $n! = n \cdot (n-1) \cdots 2 \cdot 1$)

```
1  Factorial:
2      addi    sp, sp, -8
3      sw      ra, 0(sp)
4      li      t0, 1
5      beq     a0, t0, last_sit
6      sw      a0, 4(sp)
7      ________________
8      ________________
9      lw      t0, 4(sp)
10     ________________
11     j       fact_done
12 last_sit:
13     ________________
14 fact_done:
15     lw      ra, 0(sp)
16     addi    sp, sp, 8
17     mv      a0, a1
18     jr      ra
```

Fill in the missing code below.

line 7: ____________

line 8: ____________

line 10: ____________

line 13: ____________

Solution:

line 7: `addi a0, a0, -1`,

line 8: `jal Factorial`,

line 10: `mul a1, t0, a1`,

line 13: `li a1, 1` or `addi a1, x0, 1`.

RISC-V Practices

Singly-linked list is a common and useful data structure. In this problem, you are going to implement a linked list operation in RISC-V assembly. Assume the assembly is for **a 32-bit machine**. Also, by convention, consecutive fields occupy consecutive bytes within the structure by their declaration order, and the first field takes the lowest address. The node in a singly-linked list is defined as a **struct** type as follows.

```c
struct Node
{
    // Value of this node
    int val;
    // Pointer to the next node
    struct Node *next_node;
};
```

```
# a0: address of node A; a1: address of node B
insert_node:
    lw t0 4(a0)


    ret
```

Then we define the function:

• insert node : Given a pointer to node A and a pointer to node B, this function will insert node B into the linked list, making node B the next node of node A. Node A is already in the list and assume that it is not the last node (tail) of the list.

RISC-V Practices

上海科技大学
ShanghaiTech University

Singly-linked list is a common and useful data structure. In this problem, you are going to implement two linked list operations in RISC-V assembly. Assume the assembly is for **a 32-bit machine**. Also, by convention, consecutive fields occupy consecutive bytes within the structure by their declaration order, and the first field takes the lowest address. The node in a singly-linked list is defined as a **struct** type as follows.

```
struct Node
{
    // Value of this node
    int val;
    // Pointer to the next node
    struct Node *next_node;
};
```

```
# a0: address of node A; a1: address of node B
insert_node:
    # temp = A->next
    lw t0 4(a0)
    # B->next = temp
    sw t0 4(a1)
    # A->next = B
    sw a1 4(a0)
    ret
```

Then we define some functions:

- insert node : Given a pointer to node A and a pointer to node B, this function will insert node B into the linked list, making node B the next node of node A. Node A is already in the list and assume that it is not the last node (tail) of the list.

RISC-V Practices

上海科技大学
ShanghaiTech University

(a) Doubly linked list is a common and useful data structure. In this problem, you are going to implement two linked list operations in RISC-V assembly. Assume the assembly is for a 32-bit machine. Also, by convention, consecutive fields occupy consecutive bytes within the structure by their declaration order, and the first field takes the lowest address. The node in a double linked list is defined as a `struct` type as follows.

```c
struct node{
        // value of this node
        int val;
        // pointer to next node
        struct node * next_node;
        // pointer to previous node
        struct node * prev_node;
};
```

Then we define some functions:

- insert_node : Given a pointer to node A and a pointer to node B, this function will insert node B into the linked list, making node B the next node of node A. Node A is already in the list and assume that it is not the last node (tail) of the list.
- switch_node : Given a pointer to node A and a pointer to node B (A and B are different and they are not adjacent), this function will exchange the location of node A and node B in the linked list without changing the node values. Assume that nodes A and B are neither the head nor the tail of the linked list, otherwise, they can be at any positions in the linked list.

Please fill in the following RISC-V codes to implement these two functions

```
// a0: address of node A; a1: address of node B
insert_node:
  lw t0 4(a0)

  ____________________

  ____________________

  ____________________

  ____________________
  ret
```

```
// a0: address of node A; a1: address of node B
switch_node:
  lw t0 4(a0)
  lw t1 4(a1)
  lw t2 8(a0)
  lw t3 8(a1)

  ____________________

  ____________________

  ____________________

  ____________________

  ____________________

  ____________________

  ____________________

  ____________________
  ret
```

RISC-V Practices

ShanghaiTech University

(a) Doubly linked list is a common and useful data structure. In this problem, you are going to implement two linked list operations in RISC-V assembly. Assume the assembly is for a 32-bit machine. Also, by convention, consecutive fields occupy consecutive bytes within the structure by their declaration order, and the first field takes the lowest address. The node in a double linked list is defined as a `struct` type as follows.

```
struct node{
        // value of this node
        int val;
        // pointer to next node
        struct node * next_node;
        // pointer to previous node
        struct node * prev_node;
};
```

Then we define some functions:

- insert_node : Given a pointer to node A and a pointer to node B, this function will insert node B into the linked list, making node B the next node of node A. Node A is already in the list and assume that it is not the last node (tail) of the list.
- switch_node : Given a pointer to node A and a pointer to node B (A and B are different and they are not adjacent), this function will exchange the location of node A and node B in the linked list without changing the node values. Assume that nodes A and B are neither the head nor the tail of the linked list, otherwise, they can be at any positions in the linked list.

Please fill in the following RISC-V codes to implement these two functions

```
// a0: address of node A; a1: address of
   node B
insert_node
// t0 for A->next_node
lw t0 4(a0)
// B->next_node = A->next_node
sw t0 4(a1)
// A->next_node = B
sw a1 4(a0)
// B->prev_node = A
sw a0 8(a1)
// B->next_node->prev_node = B
sw a1 8(t0)
ret
```

RISC-V Practices

上海科技大学 ShanghaiTech University

(a) Doubly linked list is a common and useful data structure. In this problem, you are going to implement two linked list operations in RISC-V assembly. Assume the assembly is for a 32-bit machine. Also, by convention, consecutive fields occupy consecutive bytes within the structure by their declaration order, and the first field takes the lowest address. The node in a double linked list is defined as a `struct` type as follows.

```
struct node{
        // value of this node
        int val;
        // pointer to next node
        struct node * next_node;
        // pointer to previous node
        struct node * prev_node;
};
```

Then we define some functions:

- insert_node : Given a pointer to node A and a pointer to node B, this function will insert node B into the linked list, making node B the next node of node A. Node A is already in the list and assume that it is not the last node (tail) of the list.
- switch_node : Given a pointer to node A and a pointer to node B (A and B are different and they are not adjacent), this function will exchange the location of node A and node B in the linked list without changing the node values. Assume that nodes A and B are neither the head nor the tail of the linked list, otherwise, they can be at any positions in the linked list.

Please fill in the following RISC-V codes to implement these two functions

```
switch_node:
// temp1 = A->next
// temp2 = B->next
// temp3 = A->prev
// temp4 = B->prev

// A->next->prev = B
// A->prev->next = B
// B->next->prev = A
// B->prev->next = A
      // B->prev = A->prev
// B->next = A->next
// A->prev = B->prev
// A->next = B->next

lw t0 4(a0)
lw t1 4(a1)
lw t2 8(a0)
lw t3 8(a1)
sw a1 8(t0)
sw a1 4(t2)
sw a0 8(t1)
sw a0 4(t3)
sw t2 8(a1)
sw t0 4(a1)
sw t3 8(a0)
sw t1 4(a0)
```

RISC-V Practices

上海科技大学 ShanghaiTech University

(a) Doubly linked list is a common and useful data structure. In this problem, you are going to implement two linked list operations in RISC-V assembly. Assume the assembly is for a 32-bit machine. Also, by convention, consecutive fields occupy consecutive bytes within the structure by their declaration order, and the first field takes the lowest address. The node in a double linked list is defined as a `struct` type as follows.

```c
struct node{
        // value of this node
        int val;
        // pointer to next node
        struct node * next_node;
        // pointer to previous node
        struct node * prev_node;
};
```

```
# a0: address of node A; a1: address of node B
insert_node:
    lw t1 4(a1)
    lw t2 8(a0)

    ret
```

Consider another senario for switch node:

- switch node : Given a pointer to node A and a pointer to node B (**A and B are different and they are adjacent, the next node of A is B**), this function will exchange the location of node A and node B in the linked list without changing the node values. Assume that nodes A and B are neither the head nor the tail of the linked list, otherwise, they can be at any positions in the linked list.

RISC-V Practices

上海科技大学 ShanghaiTech University

(a) Doubly linked list is a common and useful data structure. In this problem, you are going to implement two linked list operations in RISC-V assembly. Assume the assembly is for a 32-bit machine. Also, by convention, consecutive fields occupy consecutive bytes within the structure by their declaration order, and the first field takes the lowest address. The node in a double linked list is defined as a `struct` type as follows.

```c
struct node{
        // value of this node
        int val;
        // pointer to next node
        struct node * next_node;
        // pointer to previous node
        struct node * prev_node;
};
```

Consider another senario for switch node:

- switch node : Given a pointer to node A and a pointer to node B (**A and B are different and they are adjacent, the next node of A is B**), this function will exchange the location of node A and node B in the linked list without changing the node values. Assume that nodes A and B are neither the head nor the tail of the linked list, otherwise, they can be at any positions in the linked list.

```
# a0: address of node A; a1: address of node B
insert_node:
     # temp1 = B->next
    lw t1 4(a1)
     # temp2 = A->prev
    lw t2 8(a0)
    # B->next->prev = A
    sw a0 8(t1)
    # A->prev->next = B
    sw a1 4(t2)
    # A->next = B->next
    sw t1 4(a0)
    # A->prev = B
    sw a1 8(a0)
    # B->next = A
    sw a0 4(a1)
    # B->prev = A->prev
    sw t2 8(a1)
    ret
```

Venus Q&A

Thank you for attending the discussion!

Wish you good luck doing homework/projects/exams!