

BÀI TẬP ÔN TẬP

LẬP TRÌNH HỆ THỐNG

Bài tập 1. Thực hiện các phép chuyển đổi và tính toán sau:

- a. Chuyển các số hexan sang hệ nhị phân:

0x39A7F8₁₆ = 2

0xD5E4C₁₆ = 2

- b. Chuyển số nhị phân sang hệ hexan (16):**

$$110010010111011_2 = \dots \quad 16$$

$$100110111001110110101_2 = \dots$$

- c. Thực hiện tính toán:

$$0x506 + 0x12 = \dots$$

0x503C - 0x42 = 16

0x6653 + 98 = 16

Bài tập 2. Byte ordering

a) Cho đoạn chương trình:

```
/* Biến val gồm 4 byte đánh thứ tự từ 1 đến 4 từ địa chỉ bắt đầu */
int val = 0x87654321;
/* pointer trả đến ô nhớ lưu trữ biến val */
byte_pointer valp = (byte_pointer) &val;
/* A. hàm trả về byte thứ 1 kể từ địa chỉ bắt đầu */
show_bytes(valp, 1);
/* B. hàm trả về byte thứ 2 kể từ địa chỉ bắt đầu */
show_bytes(valp, 2);
```

Kết quả trả về của 2 hàm **show_bytes()** sẽ khác nhau như thế nào trong trường hợp chạy trên hệ thống sử dụng little-endian và big-endian?

| Hệ thống | <code>show_bytes(valp,1)</code> | <code>show_bytes(valp,2)</code> |
|---------------|---------------------------------|---------------------------------|
| Little-endian | | |
| Big-endian | | |

b) Cho các byte đang nằm trong một số ô nhớ có địa chỉ tương ứng như bên dưới

Mỗi ô tương ứng với 1 byte trong bộ nhớ. Biết hệ thống đang xét là Linux 32 bit.

| | | | | | | | | | |
|----------------|------------|------------|-------------|-------------|------------|-------------|-------------|-------------|-------------|
| <i>Giá trị</i> | 0x1 | 0x2 | 0x56 | 0xAB | 0x0 | 0x12 | 0x23 | 0x16 | 0x10 |
| <i>Địa chỉ</i> | 0x102 | 0x103 | 0x104 | 0x105 | 0x106 | 0x107 | 0x108 | 0x109 | 0x10A |

Cho trước địa chỉ lưu trữ của một số biến sau, xác định giá trị cụ thể của chúng?

| Biến | Địa chỉ | Giá trị |
|----------|---------|---------|
| char a | 0x10A | 0x10 |
| int b | 0x104 | |
| short c | 0x106 | |
| double d | 0x102 | |
| char * e | 0x106 | |

c) Cho các biến có giá trị và địa chỉ như bên dưới

| Biến | Địa chỉ | Giá trị (hệ 10) |
|---------|---------|-----------------|
| int a | 0x200 | -10 |
| short b | 0x206 | 258 |
| char c | 0x208 | 'e' |

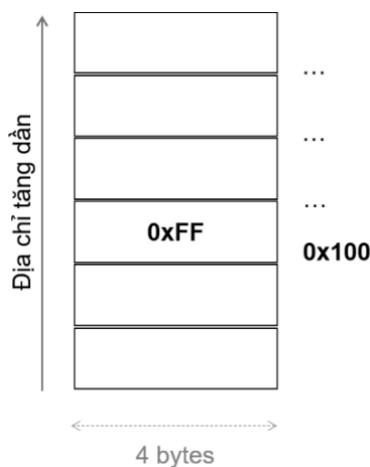
Hãy điền các giá trị tương ứng vào các ô nhớ bên dưới?

| | | | | | | | | | |
|---------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Giá trị | <input type="text"/> |
| Địa chỉ | 0x200 | 0x201 | 0x202 | 0x203 | 0x204 | 0x205 | 0x206 | 0x207 | 0x208 |

Bài tập 3. Giả sử có các giá trị sau đang được lưu trong các ô nhớ và các thanh ghi:

| Địa chỉ | Giá trị | Thanh ghi | Giá trị |
|---------|---------|-----------|---------|
| 0x100 | 0xFF | %eax | 0x100 |
| 0x104 | 0xAB | %ecx | 0x1 |
| 0x108 | 0x13 | %edx | 0x3 |
| 0x10C | 0x11 | | |

a) Hãy điền vào hình bên dưới các địa chỉ và giá trị tương ứng của các ô nhớ.



b) Xác định giá trị lấy được ở lệnh mov

Giả sử cho lệnh **movl [toán hạng x], %ebx** để lấy giá trị với toán hạng x và gán cho thanh ghi %ebx. Dựa vào các giá trị ở trên, điền các giá trị sẽ lấy được với các toán hạng sau:

| Toán hạng x | Giá trị lấy được |
|-----------------|---|
| %eax | Giá trị lưu trong thanh ghi eax: 0x100 |
| 0x104 | |
| \$0x108 | |
| (%eax) | |
| 4(%eax) | |
| 9(%eax, %edx) | |
| 0xFC(%ecx, 4) | |
| (%eax, %edx, 4) | |

Lưu ý: Câu c và d bên dưới có giá trị các thanh ghi/ô nhớ ở mỗi câu lệnh lấy từ bảng trên, các lệnh thực thi riêng biệt không liên quan đến nhau, hệ thống đang xét là 32bit.

c) Tác động của một số lệnh

Điền vào chỗ trống thanh ghi/ô nhớ nào bị thay đổi giá trị và giá trị mới là bao nhiêu?

| Câu lệnh | Thanh ghi/ô nhớ bị thay đổi | Giá trị mới |
|------------------------------------|-------------------------------|---------------------------|
| addl %ecx, (%eax) | Ô nhớ có địa chỉ 0x100 | 0xFF + 0x1 = 0x100 |
| imull \$16, (%eax, %edx, 4) | Ô nhớ có địa chỉ ... | |
| subl %edx, %eax | | |
| movl (%eax, %edx, 4), %eax | | |
| leal (%eax, %edx, 4), %eax | | |

d) Viết một số lệnh hiện thực các ý tưởng sau

| Tác vụ | Lệnh assembly |
|---|----------------------------|
| Tăng giá trị thanh ghi %eax lên 1 đơn vị | incl %eax / addl \$1, %eax |
| Nhân giá trị đang lưu trong %edx với 8 | |
| Tính toán địa chỉ ở ô nhớ nằm phía trên địa chỉ đang lưu trong %eax 20 byte. Kết quả lưu trong %ecx | |
| Trừ giá trị của %eax cho giá trị trong ô nhớ có địa chỉ 0x104, lưu kết quả vào thanh ghi đó. | |
| Tắt 4 bit thấp nhất của giá trị %edx, các bit còn lại giữ nguyên. | |
| Chỉ lấy 2 byte từ ô nhớ nằm trên địa chỉ đang lưu trong %eax 4 byte, kết quả lưu trong thanh ghi 2 byte của %ecx. | |

Bài tập 4. Cho trước các giá trị đang lưu tại các địa chỉ và thanh ghi tương ứng:

| Địa chỉ | Giá trị | Thanh ghi | Giá trị |
|---------|---------|-----------|---------|
| 0x100 | 0x1 | %eax | 0x100 |
| 0x104 | 0x2 | %ecx | 0x1 |
| 0x108 | 0x4 | %edx | 0x3 |
| 0x10C | 0x1234 | | |

Cho đoạn mã assembly bên dưới:

```

1  movl  8(%eax), %ebx
2  leal   (%eax, %ebx), %ebx
3  addl   (%ebx), %ecx
4  imull  %ecx, %edx
5  movb   %dl, 12(%eax)
6  movl   %ecx, 4(%eax)
7  movl   %ebx, 8(%eax)
8  movl   %eax, (%eax)
9  xorl   %eax, %eax

```

Sau khi thực hiện đủ 9 lệnh trên, xác định giá trị mới tại các địa chỉ ô nhớ và thanh ghi tương ứng dưới bảng sau:

| Địa chỉ/Thanh ghi | Giá trị mới | Giải thích |
|-------------------|-------------|------------|
| 0x100 | | |
| 0x104 | | |
| 0x108 | | |
| 0x10C | | |
| %eax | | |
| %ebx | | |
| %ecx | | |
| %edx | | |

Bài tập 5. Giả sử 1 lập trình viên muốn viết 1 đoạn mã thực hiện chức năng sau:

```

1 int req_fun(int a, int b){
2     int val = 0;
3     int x = a & 0xFFFF0000;
4     int y = b & 0xFFFF;
5     val = x | y;
6     return val;
7 }

```

a. Hoàn thành đoạn mã assembly bên dưới để có chức năng tương tự, kết quả val lưu trong %eax.

```

1  movl  8(%ebp), %ebx
2  movl  12(%ebp), %ecx
3  andl  $0xFFFF0000, .....
4  andl  ..... , .....
5  ..... , .....
6  ..... , .....

```

b. Một lập trình viên khác có ý tưởng khác với đoạn code tối ưu hơn, hãy hoàn thành đoạn code assembly bên dưới?

```

1  movl  8(%ebp), %ebx
2  movl  12(%ebp), %ecx
3  movl  ..... , %eax
4  movw  ..... , %ax

```

Bài tập 6. Cho đoạn mã assembly như bên dưới:

x, y, z là các tham số thứ nhất, thứ 2 và thứ 3 của hàm arith

```

1  movl 8(%ebp), %ecx
2  movl 12(%ebp), %eax
3  imull %ecx, %eax
4  subl %ecx, %eax
5  leal  (%eax,%eax,4), %eax
6  addl 16(%ebp), %eax
7  sarl $2, %eax

```

Dựa vào mã assembly, điền vào những phần còn trống trong các hàm C tương ứng:

a. Hàm arith() phiên bản 1

```

1  int arith(int x, int y, int z)
2  {
3      int t1 = .....;
4      int t2 = .....;
5      int t3 = .....;
6      int t4 = .....;
7      int t5 = .....;
8      return t5;
9  }

```

b. Hàm arith() phiên bản 2 (rút gọn)

```

1  int arith(int x, int y, int z)
2  {
3      int t1 = .....;
4      return t1;
5  }

```

Bài tập 7. Cho một chuỗi số str, một ký tự số n và 1 đoạn mã assembly xử lý như bên dưới:

```

1  movl $str, %eax          # $str: address of str
2  xorl %ebx, %ebx
3  xorl %ecx, %ecx
4  movb (%eax), %bl
5  subl $48, %ebx
6  imull $100, %ebx
7  movb 1(%eax), %cl
8  subl $48, %ecx
9  imull $10, %ecx
10  addl %ecx, %ebx
11  xorl %ecx, %ecx
12  movb 2(%eax), %cl
13  subl $0x30, %ecx
14  addl %ecx, %ebx
15  xorl %eax, %eax
16  movb (n), %al
17  subl $0x30, %eax
18  andl %ebx, %eax      # return

```

a. Hoàn thành đoạn mã C tương ứng với đoạn mã assembly trên.

```

1 int func(char* str, char n) {
2     .....
3     .....
4     .....
5     .....
6     .....
7     return ....;
8 }
```

b. Ý nghĩa các lệnh xorl ở dòng 2, 3, 11, 15? Nếu không có thì tác động gì có thể xảy ra?

.....
.....

c. Dự đoán chức năng của đoạn mã trên?

.....
.....

d. Giả sử str = ‘125’, n = ‘7’. Xác định giá trị được trả về ở cuối hàm?

.....
.....

Bài tập 8. Cho đoạn mã assembly của hàm test dưới đây được tạo bởi GCC:

x, y lần lượt là các tham số thứ nhất và thứ hai của hàm test

```

1      movl 8(%ebp), %eax
2      movl 12(%ebp), %edx
3      cmpl $-3, %eax
4      jge .L2
5      cmpl %edx, %eax
6      jle .L3
7      imull %edx, %eax
8      jmp .L4
9 .L3:
10     leal (%edx,%eax), %eax
11     jmp .L4
12 .L2:
13     cmpl $2, %eax
14     jg .L5
15     xorl %edx, %eax
16     jmp .L4
17 .L5:
18     subl %edx, %eax
19 .L4:
20     // return val
```

- a. Hoàn thành đoạn mã C tương ứng với đoạn mã assembly trên, trong đó giá trị cuối cùng của **val** được lưu trong **%eax** để trả về tại **.L4**.

(Bài tập có nhiều đáp án, SV chỉ cần ánh xạ đúng các điều kiện và đoạn code tương ứng)

```

1 int test(int x, int y) {
2     int val = .....;
3     if ( ..... ) {
4         if ( ..... )
5             val = .....;
6     else
7         val = .....;
8 } else if ( ..... )
9     val = .....;
10    return val;
11 }
```

- a. Giả sử với tham số **x = 4, y = 2**. Xác định giá trị **val** và giải thích?

.....
.....
.....

- b. Giả sử với tham số **x = 1, y = 9**. Xác định giá trị **val** và giải thích?

.....
.....
.....

Bài tập 9. Cho đoạn mã assembly như bên dưới:

x là tham số thứ nhất của hàm fun_b

```

1      movl 8(%ebp), %ebx
2      movl $0, %eax          # val
3      movl $0, %ecx          # i
4      jmp   .L2
5 .L1:
6      leal  (%eax,%eax), %edx
7      movl %ebx, %eax
8      andl $1, %eax
9      orl   %edx, %eax
10     shr1  %ebx
11     addl $1, %ecx
12 .L2:
13     cmpl $5, %ecx
14     jle  .L1
```

- a. Hoàn thành đoạn mã C tương ứng với đoạn mã assembly trên. Biết giá trị cuối cùng của **val** lưu trong **%eax** để trả về sau khi thoát vòng lặp **for**.

```

1 int fun_b(unsigned x) {
2     int val = 0;
3     int i;
4     for (.....) {
5         .....
6         .....
7     }
8     return val;
9 }
```

- b. Với **x = 32**, xác định giá trị của **val**? Giải thích?
-
.....
.....

Bài tập 10. Cho hàm C như sau:

```

1 int my_function()
2 {
3     int first_var = 0;
4     int second_var = 0xdeadbeef;
5     char str[2] = ?>;
6
7     char buf[10];
8     gets(buf);
9     return len(buf);
10 }
```

GCC tạo ra mã assembly tương ứng như sau:

```
.section .data
.LC0:
.byte 0x68,0x69,0x74,0x68,0x75,0x0 # mảng các byte liên tục nhau

.section .text
1 my_function:
2     pushl %ebp
3     movl %esp, %ebp
4     subl $24, %esp
5     movl $0, -4(%ebp)
6     movl $0xdeadbeef, -8(%ebp)
7     movw (.LC0), %dx
8     movw %dx, -12(%ebp)
9     leal -24(%ebp), %eax
10    pushl %eax
11    call gets
```

```

12  leal    -24(%ebp), %eax
13  pushl   %eax
14  call    len
15  leave
16  ret

```

Giả sử hàm **my_function** bắt đầu thực thi với những giá trị thanh ghi như sau:

| Thanh ghi | Giá trị |
|-----------|----------|
| %esp | 0x800168 |
| %ebp | 0x800180 |

Biết **.LC0** là label của 1 vùng nhớ.

- a. Giá trị của thanh ghi **%ebp** sau khi thực thi dòng lệnh assembly thứ 3? Giải thích.

.....
.....
.....
.....
.....

- b. Giá trị của thanh ghi **%esp** sau khi thực thi dòng lệnh assembly thứ 4? Giải thích.

.....
.....
.....
.....
.....

- c. Hàm **my_function** có 1 biến cục bộ **str**, là 1 mảng char gồm 2 ký tự. Quan sát mã assembly, hãy cho biết 2 ký tự được gán cho mảng **str** là gì?

.....
.....
.....
.....
.....
.....
.....
.....
.....

- d. Xác định địa chỉ cụ thể của vị trí sẽ lưu chuỗi input nhận về từ hàm **gets()**? Giải thích?

.....
.....
.....
.....
.....
.....
.....
.....

- e. Giả sử khi gọi **gets** ở dòng code assembly thứ 11, nhận được 1 chuỗi “Hello NT209”. Vẽ stack frame của **my_function** ngay sau khi hàm **gets** trả về.
Lưu ý: Cần chú thích địa chỉ, giá trị của các ô nhớ trong stack frame của **my_function**, bao gồm cả các ô nhớ chứa biến cục bộ, tham số và chuỗi đã nhập với **gets**.

- f. **gets** không giới hạn độ dài chuỗi mà nó nhận. Dựa vào stack frame của **my_function** đã vẽ, hãy tìm độ dài tối đa của **buf** (số ký tự) có thể nhập sao cho khi nhập vẫn chưa ghi đè lên bất kỳ ô nhớ quan trọng nào trong stack của **my_function** để bị lỗi segmentation fault?

- g. Chương trình có thể có lỗi hỏng buffer overflow. Thủ tìm một chuỗi buf sao cho có thể ghi đè lên biến cục bộ **second_var** một giá trị mới là **0x1234ABCD**.

Bài tập 11. Trong hệ thống 32 bit, cho mảng **T A[N]** với **T** và **N** chưa biết. Biết tổng kích thước của mảng A là **28 bytes**, T là 1 kiểu dữ liệu cơ bản.

- a. Xác định T và N ? Giải thích? Liệt kê tất cả các trường hợp thỏa mãn và dạng khai báo của mảng A với mỗi T và N tìm được.

- b. Giả sử với 1 trường hợp của **T A[N]** ở trên, ta có đoạn mã assembly tương ứng truy xuất các phần tử của mảng như bên dưới (? là giá trị chưa biết)

```
1      movl  $A, %eax // address of A
2      movw  $0, (%eax)
3      movl  $1, %ecx // chỉ số i
4 .L1:
5      xorl  %ebx, %ebx
6      movw  -2(%eax, %ecx, 2), %bx
7      addl  %ecx, %ebx
8      movw  %bx, (%eax, %ecx, 2)
9      incl  %ecx
10     cmpl  ____, %ecx
11     jl    .L1
```

- Xác định mảng $T A[N]$ đang được dùng trong đoạn mã assembly trên? Giải thích? Xác định giá trị ? còn thiếu?

.....
.....
.....
.....

- Hãy hoàn thành đoạn code C có chức năng tương ứng với đoạn mã assembly?

```
1 ..... A[.....]; // khai báo mảng A
2 A[0] = ....;
3 for (int i = .....; i < .....; i++)
4 {
5 .....  
6 .....  
7 .....  
8 .....  
9 }
```

c. Với đoạn mã và **T A[N]** ở câu b, cho biết sau khi thực hiện các đoạn code trên, ta thu được mảng A với các giá trị phần tử như thế nào?

.....
.....
.....

d. Vẫn với mảng **T A[N]** ở câu b, cho địa chỉ của **A[0]** là **0x1010**. Cho biết địa chỉ của phần cuối tử của mảng A?

.....
.....
.....

Bài tập 12. Cho mảng T A[R][C] với T là kiểu dữ liệu cơ bản, R và C chưa biết.

Xác định T, R và C của mảng A trong các trường hợp sau và viết dạng khai báo tương ứng.

a. Trường hợp 1: A là 1 ma trận vuông, mỗi mảng con nằm cách nhau 20 bytes, kích thước 1 phần tử lớn hơn 1 byte và không tồn tại phần tử A[6][6].

b. Trường hợp 2: Biết địa chỉ của A là 0x100C, địa chỉ của phần tử A[2][2] là 0x1020 và tổng kích thước của mảng A là 40 bytes.

c. Trường hợp 3: Tổng kích thước của A khác nhau trên hệ thống 32 bit và 64 bit, với chênh lệch kích thước là 96 bytes và trong mảng tồn tại phần tử A[4][3].

Bài tập 13. Cho 2 đoạn mã C và assembly có chức năng tương đương, trong đó giá trị N chưa biết.

Biết địa chỉ mảng A là tham số thứ nhất, giá trị val là tham số thứ 2

Code C

```
1 # define N ?
2 void matrix_set_val(int A[N][N], int val)
3 {
4     int i;
5     for (i = 0 ; i < N; i++)
6         A[i][i] = val;
7 }
```

Code assembly

```
1      movl  8(%ebp), %ecx
2      movl  12(%ebp), %edx
3      movl  $0, %eax
4      .L14:
5      movl  %edx, (%ecx,%eax)
6      addl  $68, %eax
7      cmpl  $1088, %eax
8      jne   .L14
```

Hãy phân tích đoạn mã C và assembly trên để xác định giá trị của N?

Page 13 of 18

Bài tập 14. Cho định nghĩa struct như bên dưới trong Linux 32 bit, có alignment.

```
1 typedef struct {  
2     int* a[2];  
3     char b;  
4     short c;  
5     double d;  
6 } str1;
```

- a. Vẽ hình minh họa việc cấp phát struct **str1** trong bộ nhớ, giả sử địa chỉ bắt đầu là p:8?

.....
.....
.....
.....

- b. Có bao nhiêu byte trống được chèn thêm vào struct khi thực hiện căn chỉnh?

.....

- c. Tổng kích thước của struct trên là bao nhiêu?

.....
.....

- d. Giả sử có code assembly của 1 hàm **func** với 2 tham số lần lượt là *i* và *val*, thực hiện xử lý các thành phần của 1 struct **str1** có tên là **s** như sau.

```

1 func:
2   movl $s, %eax           # address of struct s
3   movq $1, 12(%eax)
4   xorl %ecx, %ecx
5   movl 12(%ebp), %ebx
6   movw 10(%eax), %cx
7   leal 1(%ebx, %ecx), %ebx
8   movl %ebx, 4(%eax)
9   movl 12(%ebp), %ebx
10  andl $0xF, %ebx
11  addb $0x30, %bl
12  movb %bl, 8(%eax)
13  movl 8(%ebp), %ebx
14  movl %ebx, (%eax, %ebx, 4)
15  movl 4(%eax), %eax      # return

```

Hoàn thành đoạn code C có chức năng tương ứng với đoạn mã assembly trên?

```

1 int func(int i, int val)
2 {
3   str1 s;
4   .....
5   .....
6   .....
7   .....
8   .....
9   return ....;
10}

```

- e. Giả sử sử dụng chung khai báo struct **str1** trên hệ thống Linux 64 bit có alignment, chênh lệnh kích thước của struct trên 2 hệ thống là bao nhiêu bytes? Giải thích?

.....

- f. Giả sử trên Linux 32bit có alignment, cần tối ưu hóa kích thước vùng nhớ được dùng để cấp phát struct **str1**, khi đó cần khai báo lại struct **str1** như thế nào?

.....

Bài tập 15. Cho struct như bên dưới trong Linux 32-bit, có alignment.

```

1 typedef struct {
2     short a[4];
3     char b;
4     int c;
5 } str2;

```

Một hàm func1 được dùng để lần lượt gán giá trị cho thành phần a[i] và c của struct, kết quả trả về là giá trị của thành phần c như bên dưới.

```

1 int func1(int i, int val)
2 {
3     str2 s;
4     s.c = 1;
5     s.a[i] = val;
6     return s.c;
7 }

```

a. Vẽ hình minh họa việc cấp phát struct **str2** trong bộ nhớ, giả sử địa chỉ bắt đầu là p:8?

.....
.....
.....

b. Tổng kích thước của struct trên là bao nhiêu?

.....
.....

c. Tìm giá trị trả về của hàm **func** với các tham số sau? Giải thích các thay đổi có trong vùng nhớ của struct?

Giả định compiler chỉ warning khi có truy xuất ngoài mảng, vẫn cho chương trình chạy bình thường.

- **func1(2, 2)**

.....
.....
.....
.....
.....
.....
.....

- **func1(4, 2)**

.....
.....
.....
.....
.....
.....
.....

- `func1(6, 2)`

Bài tập 16. Cho 2 định nghĩa struct với 2 giá trị A và B chưa biết.

```
1 typedef struct {
2     short x[A][B];          /* Hàng số A và B chưa biết */
3     int y;
4 } str1;
5
6 typedef struct {
7     char array[B];        /* Hàng số B chưa biết */
8     int t;
9     short s[B];
10    int u;
11 } str2;
```

Cho đoạn code C cùng đoạn mã assembly tương ứng như bên dưới:

```
1 void setVal(str1 *r1, str2 *r2)
2 {
3     int v1 = r2->t;
4     int v2 = r2->u;
5     r1->y = v1+v2;
6 }
```

```
1 setVal:  
2     movl 12(%ebp), %eax  
3     movl 36(%eax), %edx  
4     addl 12(%eax), %edx  
5     movl 8(%ebp), %eax  
6     movl %edx, 92(%eax)
```

Dựa vào tương quan giữa 2 đoạn mã C và assembly, xác định 2 giá trị **A** và **B**, biết hệ thống 32 bit và có yêu cầu alignment.

Bài tập 17. Cho 2 file main.c và fib.c như sau.

| | |
|---|--|
| <pre>/* main.c */ 1. void fib (int n); 2. int main (int argc, char** argv) 3. { 4. int n = 0; 5. sscanf(argv[1], "%d", &n); 6. fib(n); 7. }</pre> | <pre>/* fib.c */ 1. #define N 16 2. static unsigned int ring[3][N]; 3. void print_bignat(unsigned int* a) 4. { 5. int i; 6. ... 7. } 8. void fib (int n) { 9. int i; 10. static int carry; 11. ... 12. }</pre> |
|---|--|

Hoàn thành bảng sau về các symbol có trong symbol table có trong 2 mô-đun main.o và fib.o, xác định các symbol là **local/global** hay **external, strong** hay **weak**.

- Ghi ‘-’ ở cả 2 cột nếu tên không có trong symbol table của mô-đun tương ứng.
- Ghi N/A ở cột **Strong hay weak** nếu loại symbol là local.

Symbol table của main.o

| Tên symbol | Loại symbol | Strong hay weak |
|------------|-------------|-----------------|
| main | | |
| fib | | |
| n | | |

Symbol table của fib.o

| Tên symbol | Loại symbol | Strong hay weak |
|--------------|-------------|-----------------|
| ring | | |
| print_bignat | | |
| fib | | |
| canary | | |