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:

0x39A7F8$_{16}$ = 0011 1001 1010 0111 1111 100($_2$

0xD5E4C$_{16}$ = 1101 0101 1110 0100 110($_2$

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

1100100101110112 = 64BB $_{16}$

1001101110011101101012 = 1373B5 $_{16}$

c. Thực hiện tính toán:

0x506 + 0x12 = 0x518

0x503C – 0x42 =

0x6653 + 98 =

Bài tập 2. Cho đoạn chương trình:

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

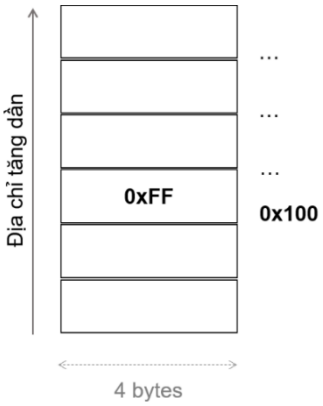
Hệ thống	show_bytes(valp,1)	show_bytes(valp,2)
Little-endian		
Big-endian		

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
0x100	0xFF
0x104	0xAB
0x108	0x13
0x10C	0x11

Thanh ghi	Giá trị
%eax	0x100
%ecx	0x1
%edx	0x3

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



b. Giả sử ta có câu lệnh **movl [toán hạng x], %ebx** để lấy giá trị dựa trên toán hạng x và đưa vào thanh ghi %ebx. Dựa vào các giá trị trong ô nhớ và thanh ghi ở trên, điền các giá trị sẽ lấy được nếu sử dụng 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)	

c. Điền vào chỗ trống ảnh hưởng của những câu lệnh dưới đây, bao gồm thanh ghi/ô nhớ nào bị thay đổi giá trị và giá trị đó là bao nhiêu?
Lưu ý: Giá trị các thanh ghi/ô nhớ ở mỗi câu lệnh vẫn lấy từ bảng trên.

Câu lệnh	Thanh ghi/ô nhớ bị thay đổi	Giá trị
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		

Bài tập 4. 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 5. Cho đoạn mã assembly như bên dưới:

x lưu tại ô nhớ (%ebp + 8), y lưu tại ô nhớ (%ebp + 12), z lưu tại ô nhớ (%ebp + 16), giá trị trả về lưu trong thanh ghi %eax

```
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 6. Cho đoạn mã assembly dưới đây được tạo bởi GCC:

x lưu tại ô nhớ (%ebp+8), y lưu tại ô nhớ (%ebp+12)

```
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. Dưới đây là 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**. Hãy điền các vị trí còn trống?

(Lưu ý: bài tập này có nhiều đáp án có thể thoả mãn đoạn code C bên dưới)

```
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  }
```

b. Giả sử với tham số **x = 4**, **y = 2**. Khi đó **val =** .....

c. Giả sử với tham số **x = 1, y = 9**. Khi đó **val =** .....

Bài tập 7. Cho đoạn mã assembly như bên dưới: *x lưu tại ô nhớ (%ebp+8)*

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

a. Dưới đây là đoạn mã C tương ứng với đoạn mã assembly. Biết giá trị cuối cùng của **val** được lưu trong **%eax** để trả về sau khi thoát vòng lặp **for**. Hãy điền vào vị trí còn trống để hoàn thiện đoạn mã bên dưới?

```
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**?

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

Bài tập 8. 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
.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 13, nhận được 1 chuỗi **"Hello world"**. 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) sao cho khi nhập vẫn chưa ghi đè lên bất kỳ ô nhớ quan trọng nào trong stack của **my_function**?

g. Chương trình có thể có lỗ hổng buffer overflow. Thử tìm một chuỗi buf sao cho có thể ghi đè lên biến cục bộ **second_var** một giá trị mới là **0xABDCEF**.

Bài tập 9. 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 C và assembly tương ứng truy xuất các phần tử của mảng như bên dưới. Hãy điền vào những chỗ còn trống ở 2 đoạn code?

Code C

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

Code assembly

```
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
```

c. Cho biết sau khi thực hiện đoạn code trên, ta thu được mảng A với các giá trị phần tử như thế nào?

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

Bài tập 10. Cho các định nghĩa sau trong code C, với giá trị N chưa biết.

```
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  }
```

Và đoạn code assembly tương ứng được tạo bởi GCC:

Địa chỉ mảng A lưu tại ô nhớ (%ebp+8), giá trị val lưu tại ô nhớ (%ebp+12)

```
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 assembly trên và xác định giá trị của N?

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

Bài tập 11. Cho struct có định nghĩa như bên dưới trong Linux 32-bit, có yêu cầu alignment.

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

Một hàm func được dùng để 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 func(int i, int val)
2  {
3        str1 s;
4        s.c = 1;
5        s.a[i] = val;
6        return s.c;
7  }
```

a. Vẽ hình minh họa việc cấp phát struct trên trong bộ nhớ?

.....

.....

.....

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 chương trình được biên dịch với compiler chỉ warning khi có truy xuất ngoài mảng, vẫn cho chương trình chạy bình thường.

- **func**(2, 2)

.....

.....

.....

- **func**(4, 2)

.....

.....

.....

- **func**(6, 2)

.....

.....

.....

Bài tập 12. Cho 2 định nghĩa struct với 2 giá trị A và B chưa biết.

```
1   typedef struct {
2       short x[A][B];          /* Hằng số A và B chưa biết */
3       int y;
4   } str1;
5
6   typedef struct {
7       char array[B];          /* Hằng 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 13. Cho 2 file **main.c** và **fib.c** như sau.

```
/* main.c */

1.  void fib (int n);
2.  int main (int argc, char** argv
      ) {
3.      int n = 0;
4.      sscanf(argv[1], "%d", &n);
5.      fib(n);
6.  }
```

```
/* fib.c */

1.  #define N 16
2.  static unsigned int ring[3][N];
3.  void print_bignat(unsigned int*
     a){
4.      int i;
5.      ...
6.  }
7.  void fib (int n) {
8.      int i;
9.      static int carry;
10.     ...
11. }
```

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		