시스템프로그램 중간시험 (2020학년도 2학기, 10/29/2020)

학과		학번		학년		이름		

☆ **답안지 각 페이지 상단에 소속학과, 학번, 학년, 이름을 작성하세요.**

(1) Fill in the blanks of the following sentences. Choose your answer in the Term-Box below.

- ○ The gcc compiler driver reads a source file and translates it into an executable object file. The translation is performed in the sequence of four phases and the programs that perform the four phases are preprocessor, compiler, assembler, and (①).
- ○ The program counter, which uses register (②) in x86-64, indicates the address in memory of the next instruction to be executed.
- ○ Some machines choose to store the object in memory ordered from least significant byte to most, while other machines store them from most to least. The latter convention, where the most significant byte comes first, is referred to as (③).
- ○ The (④) is a technique that allows a single CPU to execute multiple flows of control. It involves having multiple copies of some of the CPU hardware, such as program counters and register files, while having only single copies of other parts of the hardware, such as the units that perform floating-point arithmetic.
- ○ To inspect the contents of machine-code files, a class of programs known as disassemblers can be invaluable. These programs generate a format similar to assembly code from the machine code. With Linux systems, the program (⑤) can serve this role given the -d command-line flag.
- ○ For a binary number x = $(x_{w-1}, x_{w-2}, \cdots, x_0)$, its 2's complement encoding can be defined as

  $B2T_w(x) = (⑥\ \ ) + \sum_{i=0}^{w-2} x_i \cdot 2^i$

- ○ IEEE FP format defines 4 different rounding modes. Among those, (⑦) is the default mode and attempts to find a closest match. For halfway values, it adopts the convention that it rounds the number either upward or downward such that the LSB of the result is zero.
- ○ In x86-64, registers %rax, %rdi, %rsi, %rdx, %rcx, and %r8~%r11 are classified as (⑧). When procedure P calls procedure Q, P must preserve the values of these registers, ensuring that they can be used by Q without any restriction.
- ○ A switch statement (in C language) provides a multi-way branching capability based on the value of an integer index. It is usually compiled with the (⑨⑩), which is an array where each entry **i** is the address of a code segment implementing the action the program should take when the switch index equals **I**.
- ○ With (⑩), different parts of a program, including text, data, stack, and heap, are loaded into different regions of memory each time the program is run.

[Term-Box]

stack randomization	symbol table	hyperthreading	round-toward-zero	readelf
ASLR	linker	piplining	round-to-even	objdump
%rsp	$-x_{w-1} \cdot 2^{w-1}$	jump table	little endian	callee saved register(s)
%rip	$-x_{w-1} \cdot (2^{w-1}-1)$	data section	big endian	caller saved register(s)

(2) Assume we are running code on a 6-bit machine using 2's-complement arithmetic for signed integers. A short integer is encoded using 3-bits. Fill in the empty boxes in the table below. You need not fill in entries marked "-". The following definitions are used in the table:

```
short sy = -4;
int y = sy;
int x = -20;
unsigned ux = x;
```

Expression	Decimal representation	Binary representation
Zero	0	000000
—	10	001010
—	-16	⑧
x + 2	①	⑨
ux	②	⑩
y + 3	③	⑪
x >> 3	④	⑫
-TMax	⑤	⑬
-TMin	⑥	⑭
TMax + TMax	⑦	⑮

(3) Consider the following 8-bit floating point representation based on the IEEE floating point format:

- There is a sign bit in the most significant bit.
- The next 3 bits are the exponent. The exponent bias is $2^{3-1} - 1 = 3$.
- The last 4 bits are the fraction.
- The representation encodes numbers of the form: $V = (-1)^s \times M \times 2^E$, where M is the significand and E is the biased exponent.

The rules are like those in the IEEE standard(normalized, denormalized, representation of 0, infinity, and NAN). Fill in the table below. Here are the instructions for each field:

- Binary: The 8 bit binary representation.
- M: The value of the significand. This should be a number of the form x or x/y, where x is an integer, and y is an integral power of 2. Examples include 2, 3/4, 7/16, etc.
- E: The integer value of the exponent.
- Value: The numeric value represented, in the same form as significand M.

Description	Binary	M	E	Value
Zero (positive)	ⓐ	ⓖ	ⓜ	+0.0
—	0 101 0011	ⓗ	ⓝ	ⓢ
Smallest denormalized (negative)	ⓑ	ⓘ	ⓞ	ⓣ
Smallest normalized (positive)	ⓒ	ⓙ	ⓟ	ⓤ
Largest normalized (positive)	ⓓ	ⓚ	ⓠ	ⓥ
—	ⓔ	ⓛ	ⓡ	7/2
Positive infinity	ⓕ	—	—	+∞

(4) For the C code having the form shown on the left side of the following table, **gcc**, run with the command-line option **–O1**, produces the code, shown on the right side of the table. Fill in the missing parts of the C code. (Note that the control structure in the assembly code does not exactly match what would be obtained by a direct translation of the C code according to the guarded-do translation rules. However, you can fill out the missing parts of the C code by understanding the relationships of the codes.)

```
long loop4(long a, long b)
{
   long result = ①_____;
   while (②_____) {
      res = ③_____;
      b = ④_____;
   }
   return result;
}
```

```
loop4:
   testq %rsi,%rsi
   jle .L42
   movq %rsi,%rax
.L41:
   imulq %rdi,%rax
   subq %rdi,%rsi
   testq %rsi,%rsi
   jg .L41
   rep; ret
.L42:
   movq %rsi,%rax
   ret
```

(5) Consider the following assembly code for a C **for** loop.

① Based on the assembly code, fill in the missing parts of the corresponding C source code. (Note: You may only use the symbolic variables **x**, **n**, **mask**, and **result** in your expressions below – Do not use register names.)

② Which registers hold program values for **mask** and **result**?

```
loop5:
   movl %esi,%ecx
   movl $1,%edx
   movl $0,%eax
   jmp .L52
.L51:
   movq %rdi,%r8
   andq %rdx,%r8
   orq %r8,%rax
   salq %cl,%rdx
.L52:
   testq %rdx,%rdx
   jne .L51
   rep; ret
```

```
long loop5(long x, int n)
{
   long result = ①_____;
   long mask;
   for (mask = ②____; mask ③____; mask = ④_____) {
      result |= ⑤_____;
   }
   return result;
}
```

[Register for **mask**]	⑥
[Register for **result**]	⑦

(6) The following code transposes the elements of an **M×M** array, where **M** is a constant defined by **#define**. When compiled with optimization level **–O1**, **gcc** generates the following code for the inner loop of the function, as shown on the right side. Answer the following questions.

```
void transpose(long A[M][M]) {
   long i, j;
   for (i=0; i<M; i++)
      for (j=0; j<i; j++) {
         long t = A[i][j];
         A[i][j] = A[j][i];
         A[j][i] = t;
      }
}
```

```
.L6:
   movq (%rdx),%rcx
   movq (%rax),%rsi
   movq %rsi,(%rdx)
   movq %rcx,(%rax)
   addq $8,%rdx
   addq $160,%rax
   cmpq %rdi,%rax
   jne .L6
```

① Which register holds a pointer to array element A[i][j]?
② Which register holds a pointer to array element A[j][i]?
③ What is the value of **M**?

(7) In the following C code on the left-side, A and B are constants defined with #define, and GCC generates the following right-side code for the function **setVal**. In this case, what are the possible values of A and B?

```
typedef struct {
   int x[A][B]; /* Unknown constants A and B */
   long y;
} strl;

typedef struct {
   char array [B];
   int t;
   short s[A];
   long u;
} str2;

void setVal(strl *p, str2 *q) {
   long v1 = q->t;
   long v2 = q->u;
   p->y = v1 + v2;
}
```

```
void setVal(str1 *p, str2 *q)
p in %rdi, q in %rsi
setVal:
   movslq 8(%rsi), %rax
   addq 32(%rsi), %rax
   movq %rax, 224(%rdi)
   ret
```

☆ 수고했습니다~.