1 Lab overview

The purpose of the lab is to help students to get familiar with inline assembly code so that they can start to write simple inline assembly code on their own.

To complete the tasks, you need to have certain preliminary knowledge of inline assembly language. The below article is a fine tutorial for beginners, after reading it, you should be prepared for the tasks.

https://www.ibm.com/developerworks/library/l-ia/

The below link is a full GCC reference for inline assembly language, read it if necessary.

https://gcc.gnu.org/onlinedocs/gcc/Using-Assembly-Language-with-C.html

2 Lab Tasks

For each of the lab task, please capture the screen-shot of the key step and include it in your lab report.

2.1 Task 1

The box below shows an incomplete program which tries to assign the value of a to the variable b. Try to complete the program by replacing question mark in red with variable names. Compile the program and run it. Explain why you do that.

```
#include <stdio.h>
int main () {
  int a=10;
  int b;
  asm volatile ("movl %1, %%eax\n\t"
                "movl %%eax, %0\n\t"
               :"=r"(???)         /* output */
               :"r"(???)          /* input */
               :"%eax"          /* clobbered register */
               );
  printf("a=%d,b=%d\n",a,b);
  return 0;
}
```

2.2 Task 2

Program List 2 shows an incomplete program which tries to set b=a+2. Try to complete the program by replacing question mark in red with variable names. Compile the program and run it. Explain why you do that.

```
#include <stdio.h>
int main () {
  int a=100;
  int b;
  asm volatile ("movl %1, %%eax\n\t"
              "addl ???, %%eax\n\t"
              "movl %%eax, %0\n\t"
             :"=r"(???)         /* output */
             :"r"(???)          /* input */
             :"%eax"          /* clobbered register */
             );
  printf("a=%d,b=%d\n",a,b);
  return 0;
}
```

2.3 Task 3

The following incomplete program tries to set a=a^b. Complete the program, compile and run it.

```
#include <stdio.h>
int main () {
  int a=8;
  int b=5;
  asm volatile ("movl %1, %%eax\n\t"
               "movl %2, %%edx\n\t"
              "??? %%edx,%%eax\n\t"
              "movl %%eax, %0\n\t"
             :"=r"(???)         /* output */
             :"r"(???),"r"(???)          /* input */
             :"%eax","%edx"          /* clobbered register */
             );
  printf("a=%d,b=%d\n",a,b);
  return 0;
}
```

2.4 Task 4

In x86 calling convention, register EDI is typically used to pass the first argument. Compile and run the following program, describe what you have observed and try to explain it.

```
#include <stdio.h>
int foo (int arg) {
  int a=0;
  asm volatile ("movl %%edi,%0\n\t"
              :"=r"(a)         /* output */
              :          /* input */
              :          /* clobbered register */
              );
  printf("a=%d,arg=%d\n",a, arg);
  return 0;
}

int main() {
  foo(5);
  return 0;
}
```

2.5 Task 5

In x86 calling convention, register ESI is typically used to pass the second argument. Use what you have learnt from Task 2.5 to complete this program so that variable b will contain the value of the second argument.

```
#include <stdio.h>
int foo (int arg1, int arg2) {
  int a=0;
  int b=0;
  asm volatile ("movl %%edi,%0\n\t"
              "movl ???,%1\n\t"
              :"=r"(a),"=r"(b)         /* output */
              :          /* input */
              :          /* clobbered register */
              );
  printf("a=%d,arg1=%d,b=%d,arg2=%d\n",a, arg1,b,arg2);
  return 0;
}

int main() {
  foo(5,6);
  return 0;
}
```

2.6 (Bonus Point) Task 6

The LEA instruction is used to load effective address to the destination operand. By using “lea (%%rip), %0”, we can store the current value of PC to an output argument. Compile and run the following program. Explain why the value of (b-a) is different between the first and the second calls to printf(), what can you infer about the width of the NOP instruction.

```
#include <stdio.h>
int main () {
  unsigned long a=0;
  unsigned long b=0;
  asm volatile ("lea (%%rip), %0\n\t"
                "lea (%%rip), %1\n\t"
               :"=r"(a),"=r"(b)        /* output */
               :         /* input */
               :"%eax"         /* clobbered register */
               );
  printf("b-a=%ld\n",b-a);
  asm volatile ("lea (%%rip), %0\n\t"
                "nop\n\t"
                "lea (%%rip), %1\n\t"
               :"=r"(a),"=r"(b)        /* output */
               :         /* input */
               :"%eax"         /* clobbered register */
               );
  printf("b-a=%ld\n",b-a);
  return 0;
}
```