Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Department of Computer Science

Computer Architecture II
CSU34021

Tutorial 2
Intel's 64-bit Assembly with C/C++

Syed Asad Alam

Document History

Rev.	Date	Comment	Author
0.1	10-10-2020	Initial Draft	SAA
0.2	12-10-2020	Defined the procedures to be implemented	SAA
1.0	17-10-2020	Tutorial 2 released	SAA

1 Learning Outcomes

This lab satisfies the following learning outcomes of the course:

LO1 Write simple x64 assembly language functions

LO2 Explain the x64 procedure calling conventions

LO3 Write programs that mix C/C++ and x64 assembly language functions

2 Exercises

2.1 Program 1

The following procedure calculates a Fibonacci number by recursion:

```
long long fibonacci_recursion(long long fin)
{
  if(fin <= 0)
    return fin;
  else if (fin == 1)
    return 1;
  else
    return fibonacci_recursion(fin-1) + fibonacci_recursion(fin-2);
}
```

2.2 Program 2

The following procedure takes a user input through scanf, calculates the sum of the input arguments and user input and prints the result while returning the sum. The user input should also be accessible from other C/C++ functions:

```
long long use_scanf(long long a, long long b, long long c)
{
  long long sum = a+b+c;
  long long inp_int;

  printf("Please enter an integer: ");
  scanf("%lld", &inp_int);

  sum = sum+inp_int;

  printf("The sum of proc. and user inputs (%lld, %lld, %lld, %lld): %lld\n",
         a,b,c,inp_int,sum);

  return sum;
}
```

The scanf function requires two arguments. The first one is the format specifier (%lld) which can be defined as a string, similar to the string needed for printf and address of this string loaded as an argument and the second argument is the address of variable in memory where it will return the user input (as shown in the "C" code by &inp_int)

2.3 Program 3

The following are two procedures, with max5 calling max to calculate its return value.

```
_int64 max(_int64 a, _int64 b, _int64 c) {
    _int64 v = a;
    if (b > v)
        v = b;
    if (c > v)
        v = c;
    return v;
}

// inp_int: The user input in Program '1'
_int64 max5(_int64 i, _int64 j, _int64 k, _int64 l)
{
  return max(max(inp_int, i, j), k, l);
}
```

Exercises

Q1 Translate the code procedure above into x64 assembly language using the basic code generation strategy outlined in lectures. You may then generate optimized versions of your code.

Q2 What is the maximum depth of the stack (in stack frames) during the calculation of fibonacci(5)? Draw a diagram showing the state of the stack at its maximum depth during the calculation of fibonacci(5).

Q3 Using Visual Studio (or similar), create a console application with files t2.h and t2.asm containing the x64 assembly language for max, fibonacci and use_scanf. Use t2Test.cpp to test max, fibonacci and use_scanf. Please note that the source code provided may need to be modified to work with the development environment you use.

Submission

Submit your answer, by combining your code files, diagram of the stack for the fibonacci(5) and a snapshot of the console window showing evidence that your program works in a single zip file clearly mentioning your name and ID, via Blackboard. The deadline is available at the course page (`https://www.scss.tcd.ie/~alams/csu34021.html`) and Blackboard.

Marks Distribution

This coursework will be marked out of 100. Translation of each program carries a maximum of 30 marks while the stack diagram carries a maximum of 10 marks.

Deadline

The deadline is: Monday, November 09, 2020 9 pm.