# CS 277 Spring 2015
# Programming Project – Sequential Y86 Simulator (120 Points)

## Assignment

You will write a Sequential Simulator for executing Y86 machine code. (The Y86 ISA and sequential processing are described in Chapters 4.1 and 4.3 in your book). It should take one Y86 object file (.yo) as input, execute the machine code and produce the following output:

```
Steps: 43
PC: 0x00000108
Status: HLT
CZ: 0
CS: 0
CO: 0
%eax: 0x0000001C
%ecx: 0x00000000
%edx: 0x00000000
%ebx: 0xFFFFFFFD
%esp: 0xFFFFFFDC
%ebp: 0x00000100
%esi: 0x00000000
%edi: 0x0000A001
```

where Steps is the number of instructions executed, PC is the value of the program counter when the program exited, Status is the value of the status register when the program exited, and CZ,CS and CO are the values of the condition flags when the program exited. The final values contained by each register should also be printed. Use the provided y86 compiler to turn Y86 assembly code (.ys file) into an ASCII representation of the machine code in hex.

```
>./yescc mycode.ys
```

If you would prefer to have your simulator work directly with binary like a real computer system, you can convert the ASCII hex representation to its binary equivalent from the command line using the xxd tool:

```
>xxd -r -p prog4.yo prog4.bin
```

would convert the ASCII hex representation of the Y86 machine code in prog4.yo and write its equivalent in binary as prog4.bin.

## Simulator Memory

Your simulator should create a virtual address space like we have covered in class this term. Both %ebp and %esp should be initialized to point to the starting address of the stack. The simulator should run with 2MB of main memory for your Y86 programs to utilize. To keep things simple you only need to support the virtual address space defined by the least significant 19 bits of a memory address. This allows you to maintain a one-to-one mapping of virtual memory addresses to physical addresses in main memory. If you are interested in earning up to **25 extra credit** points on the project, then you can implement a 2nd simulator that supports full 32-bit addressing and performs the appropriate *virtual address translation* between virtual addresses and physical addresses in main memory (we will covering this topic towards the end of the term). If you elect to do the extra credit, you must submit both implementations for grading.

## Due Dates and Submissions

**Wednesday April 15th 11:59pm:** Email me with your intention to work either alone or with one other student from the class. If you are working in pairs, only one of you needs to send me an email. (20 Points)

**Thursday May 7th, 11:59pm:** Submit your final solution to me via email. (100 Points )