PROCEDURE

Lecturers: ARD, SWJ Lab Tutors: VDE, WUL

A procedure is a set of instructions that perform a specific task. They are invoked from another procedure — the calling procedure (or program) — and provide results to the calling program at the end of execution. Procedures (also called subroutines) are utilized primarily for routines that are called frequently by other procedures. The procedure routine is written only once, but used repeatedly; thereby, saving storage space. Procedures permit a program to be coded in modules; thus, making the program easier to code and test. A program can contain many procedures.

Defining a Procedure

NASM has very sparse, very clean, very simple syntax. To make a procedure, you just give it a label, and call the label. The call pushes the return address, and a ret pops it off and jumps to it.

Sample:

```
_main:
   mov eax, 3
   mov ebx, 5
   call _Add_Two_Things
   ; Result in eax
   ret
_Add_Two_Things:; Into eax
   add eax, ebx
   ret
```

CALL and RET Instructions

The CALL instruction calls a procedure by directing the processor to begin execution at a new memory location. The procedure uses a RET (return from procedure) instruction to bring the processor back to the point in the program where the procedure was called. Mechanically speaking, the CALL instruction pushes its return address on the stack and copies the called procedure's address into the instruction pointer. When the procedure is ready to return, its RET instruction pops the return address from the stack into the instruction pointer. In 32-bit mode, the CPU executes the instruction in memory pointed to by EIP (instruction pointer register). In 16-bit mode, IP points to the instruction.

The syntax for a CALL instruction is shown below.

```
CALL procedure_name_label
```

Tasks

1. Write an assembly language program that uses a procedure to multiply two single-digit operands. Enter several operands for the multiplicand and multiplier and display the products.
2. Write an assembly language program that uses a procedure to obtain the area of a triangle from two integers that are entered from the keyboard. Enter several sets of single-digit numbers for the base and height and display the areas.
3. Write an assembly language program that uses a procedure to convert an 8-bit binary code number to the corresponding Gray code number. The binary number is entered from the keyboard.
4. Write an assembly language procedure to find the minimum and maximum elements in an array of 32-bit integers.