Name: KEY **Id#**

COE 205, Term 082

Computer Organization & Assembly Programming

Quiz# 5

Date: Saturday, May 30, 2009

Q1. Write a macro, **movm**, that allows the movement of a memory variable to a memory variable. It is assumed that both memory variables are of the same type and can be only of type word or dword. The macro should not disturb the content of any of the registers. For example to move the content of double word variable I to double word variable J, we write movm J, I.

```
MOVM   MACRO J, I
          PUSH I
          POP  J
        ENDM
```

Q2. Discuss the advantages and disadvantages of macros in comparison to procedures.

Using macros results in faster execution of the code. However, macros result in increased memory space due to macro expansions. Procedures save space, as only one copy of the procedure is kept. However, procedure invocation overhead (to pass parameters via the stack and for call/ret) increases the execution time. Parameter passing in a macro invocation is similar to that in a procedure call of a high-level language. Note that macro invocation causes assembly-time overhead but not run-time overhead.

Macros are also useful in defining macro-instructions that extend the instruction set of a processor. There are situations which can be done only with macros and cannot be done with procedures like saving a selective set of registers on the stack.

Q3. Given the following macro definition:

```
MyMacro   MACRO  N
            M=1
            F=1
            REPT N
              DW F
              M=M+1
              F=M*F
            ENDM
          ENDM
```

Determine the code generated due to the following declaration:

```
Array  LABEL  WORD
MyMacro   5
```

```
Array  DW    1
       DW    2
       DW    6
       DW    24
       DW    120
```

Q4. Given the following macro definition:

```
SREG  MACRO  REGS
               IRP D, <REGS>
                  PUSH D
               ENDM
             ENDM
```

Determine the code generated due to the following declaration:

```
SREG < EAX, ECX, ESI>
```

```
PUSH EAX
PUSH ECX
PUSH ESI
```