Test 2 Review

Coding test only

- Write a subroutine/function/Macro that does …
- No theoretical questions
- No questions like find the output of this code
- Only write the subroutine or macro code
- Don't call the subroutine or use the macro in the main program
- Don't define variables, or use .DSEG , .CSEG
- No need to comment
- Answer all problems
- Don't write bad syntax code

Procedures:

- Every procedure should have a `name:` at the beginning and RET at the end
- the problem will identify the input arguments and the output arguments
- Input/output arguments can be:
    - Values passed though registers
    - Addresses passed through registers (input only)
    - Some applications have no return arguments
- Example of values as input and values as output: write a procedure that adds 2 words values and returns as word result. The 2 word values are provided through R1:R0 and R3:R2. The procedure return the result through R5:R4.
    - Input arguments are: R0,R1,R2,R3
    - Output arguments are: R4, R5
    - Implementation:

```
Add2W:
        MOV R5:R4, R3:R2
        ADD R4,R0
        ADC R5,R1
        RET
```

    - Note that there are no internal registers in the above code, ie no need to push/pop
- Example of no return value: write a procedure that would sum 2 words variables and store it in another word variable. The addresses of the input arguments are passed through registers X and Y and the address of the sum variable is passed through register Z. Assume all variables are in the DM.
    - Input arguments: X, Y, Z
    - Output arguments: none
    - All other registers used inside the procedure are internal registers that must be pushed/popped:
        - Push the internal registers at the beginning
        - Pop the registers write before the RET in reverse order
    - Implementation:

```
Add2W:
        PUSH  R0        ;internal regs
        PUSH  R1        ;internal regs
        PUSH  R4        ;internal regs
        PUSH  R5        ;internal regs
```

```
LD R0, X+
LD R1, X    ;1ST WORD R1:R0
LD R4, Y+
LD R5, Y    ;2ND WORD R5:R4
ADD R4,R0
ADC R5,R1
ST Z+, R4
ST Z, R5
POP R5
POP R4
POP R1
POP R0
RET
```

macros:

- Macro begins with `.MACRO name` and ends with `.ENDM`
- the problem will show a "usage" of the macro to identify the input/output arguments
- Arguments are @0-@9
- Arguments can be variables, registers, or values
- Push/pop internal registers
- Example arguments as values: write a macro that adds 2 constant values and returns them in a register. Example usage: add2 r1, 5, 6 ;r1=5+6
    - @0 = return register
    - @1, @2 = constant values
    - Implementation:

```
.MACRO   Add2
         PUSH R16       ;internal regs
         PUSH R17       ;internal regs
         LDI R16, @1
         LDI R17, @2
         CLR @0
         ADD @0,R16
         ADD @0,R17
         POP R17
         POP R16
.ENDM
```

- Example of variables as arguments: write a macro that adds 2 word variables and stores the result in a third variable. Example usage: add2w resultw, var1w, var2w ;result = var1w + var2w. Assume all variables are in DM.
    - @0 = result variable
    - @1,@2 = variables to be added
    - Implementation:

```
.MACRO   Add2w
         PUSH R0     ;internal regs
         PUSH R1     ;internal regs
         PUSH R4     ;internal regs
         PUSH R5     ;internal regs
         PUSH XL     ;internal regs
         PUSH XH     ;internal regs
```

```
        PUSH YL     ;internal regs
        PUSH YH     ;internal regs
        PUSH ZL     ;internal regs
        PUSH ZH     ;internal regs

        LDI XH, HIGH(@1)
        LDI XL, LOW(@1)      ;X->1ST VARIABLE

        LDI YH, HIGH(@2)
        LDI YL, LOW(@2)      ;Y->2ND VARIABLE

        LDI ZH, HIGH(@0)
        LDI ZL, LOW(@0)      ;Z->RESULT VARIABLE

        LD R0, X+
        LD R1, X    ;1ST WORD R1:R0
        LD R4, Y+
        LD R5, Y    ;2ND WORD R5:R4
        ADD R4,R0
        ADC R5,R1
        ST Z+, R4
        ST Z, R5


        POP ZH
        POP ZL
        POP YH
        POP YL
        POP XH
        POP XL
        POP R5
        POP R4
        POP R1
        POP R0

.ENDM
```