LAB #5: SUBROTINAS

1. MOTIVAÇÃO & OBJECTIVOS

Neste laboratório vamos introduzir o conceito de subrotinas (também chamadas funções ou procedimentos) na programação do processador didático P3 em linguagem Assembly (ASM).

O trabalho de casa para a semana 5 é a resolução de todos os exercícios em grupos de 2, devendo a solução ser entregue em papel no início da aula de laboratório de cada grupo.

2. SUBROTINA BÁSICA

O primeiro programa mostra a chamada a uma subrotina para manipular o valor de um registo.

```
 1: ; Programa lab5-1.as
 2:
 3: ; ZONA I: Definicao de constantes
 4:
 5: ; ZONA II: definicao de variaveis
 6: ; Diretivas : WORD - palavra (16 bits)
 7:              ORIG    8000h
 8: Var1         WORD     1
 9:
10: ; ZONA III: codigo
11: ; conjunto de instrucoes Assembly do programa
12:              ORIG    0000h
13:              JMP     Inicio
14:
15: ; ConvAscii: Rotina para converter dígitos em binário em ASCII.
16: ;       Entradas: R1 - palavra com o digito a processar
17: ;       Saidas: R1 – resultado processado
18: ;       Efeitos: altera R1
19: ConvAscii:   ADD    R1, '0'
20:              RET
21:
22: Inicio:      MOV     R1, M[Var1]
23: ProcWord:    CALL    ConvAscii       ; Converte o valor do digito em caracter ASCII
23:              MOV     M[Var1], R1
25: Fim:         BR      Fim
```

a) Indique quais os valores do registo R1 antes e depois da chamada da subrotina “ConvAscii”, e explique a lógica do funcionamento da subrotina.

b) Indique qual a sequência de instruções (número da linha) que é executada.

2. PASSAGEM DE PÂRAMETROS EM SUBROTINAS E SUBROTINAS RECURSIVAS

O seguinte código mostra a implementação da subrotina fatorial em assembly.

```
; Programa lab5-1.as

; ZONA I: Definicao de constantes
SP_INICIAL      EQU     FDFFh

; ZONA II: definicao de variaveis

; ZONA III: codigo
;        conjunto de instrucoes Assembly do programa
            ORIG    0000h
            JMP     Inicio

fatorial:   CMP  M[SP+2], R0     ; N==0?
            BR.NZ recurs         ; se não, salta
            MOV  R1, 1           ; valor de retorno 1
            MOV  M[SP+3], R1     ; escreve-o na pilha
            RETN 1               ; liberta um param. entrada
recurs:     MOV  R1, M[SP+2]     ; obtem parametro n
            DEC  R1              ; decrementa n
            PUSH R0              ; reserva espaço p/valor ret
            PUSH R1              ; coloca paramet. de entrada
            CALL fatorial
            POP  R1              ; obtem resultado
            MOV  R2, M[SP+2]     ; obtem N
            MUL  R1, R2          ; multiplica (resultado em R2)
            MOV  M[SP+3], R2     ; coloca result. na pilha
            RETN 1               ; liberta um param. entrada


Inicio:      MOV     R7, SP_INICIAL
             MOV     SP, R7
            PUSH R0
            PUSH 2
            CALL fatorial
            POP  R7
Fim:         BR      Fim
```

Desenhe o conteúdo do stack após a segunda invocação recursiva da função fatorial. Inspecione a memória do simulador de forma a confirmar a resposta anterior. (Imprima uma captura de écran indicando a zona do simulador que confirma esta resposta.)

🛠Exercício: Implemente o cálculo da função Fibonacci recursivamente. Use o seguinte programa em Python como exemplo:

```
def fib(N):
    if N <= 1:
        return 1
    return fib(N - 1) + fib(N - 2)


fib(5)
fib(8)
```