DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2023)

Actividad práctica 1

Sección 4 - Pauta de evaluación

Pregunta 1: Explique el código (1 ptos.)

Detalle, basándose en los nombres de las variables y *labels*, lo que realiza el siguiente fragmento de código desarrollado en el Assembly del computador básico visto en clases:

```
DATA:
  n 3
  x_value 6
  f_x     0
CODE:
  loop:
    MOV A,(n)
    CMP A,0
    JEQ end
    SUB A,1
    MOV (n),A
    MOV A,(x_value)
    SHL A,A
    MOV (x_value),A
    JMP loop
  end:
    MOV A,(x_value)
    MOV (f_x),A
```

No es necesario que explique lo que realiza cada instrucción del programa. Basta con que detalle el objetivo general de este. En esta pregunta no se otorgará puntaje parcial.

Solución: En este caso, se observa que se realiza la operación `SHL` una cantidad n de veces. Si tomamos la variable `f_x` como $f(x)$, entonces el fragmento de código es equivalente a $f(x) = x \times 2^n$. No es necesario que se explicite la función, pero sí que se señale que el valor de x se multiplica n veces por 2.

Pregunta 2: Arregle el código (2 ptos.)

El siguiente programa, desarrollado en el Assembly del computador básico visto en clases, **debería computar el cuadrado de la variable `x_value` y guardar el resultado en `x_square`**:

```
DATA:
  x_value 7
  x_square 0
CODE:
  loop:
    MOV A,(x_value)
    MOV B,(x_square)
    ADD (x_square)
    CMP A,0
    JEQ end
    SUB A,1
    MOV (x_value),A
    JMP loop
  end:
```

No obstante, si compila y ejecuta el código, se dará cuenta que no entrega el resultado esperado. Busque el error y, una vez encontrado, arréglelo para que el fragmento anterior entregue el resultado esperado. Suba el código completo como respuesta. Se descuenta **1 pto.** si encuentra el error y trata de arreglarlo sin éxito (lo que debe estar comentado en el código); y no se otorga puntaje si el programa no compila o se sube sin arreglo alguno.

Solución: El problema consiste en que el código utiliza el valor de `x_value` como sumador y como contador, por lo que cambia el valor incrementado en cada iteración. En el fragmento de código, se suma de 1 a 7 y no siete veces 7 como se espera. Por lo tanto, esto se puede resolver incluyendo una nueva variable de conteo:

```
DATA:
  x_value 7
  counter 0
  x_square 0
CODE:
  MOV A,(x_value)
  MOV (counter),A
  loop:
    MOV A,(counter)
    CMP A,0
    JEQ end
    SUB A,1
    MOV (counter),A
    MOV A,(x_value)
    MOV B,(x_square)
    ADD (x_square)
    JMP loop
  end:
```

No importa cómo se haya resuelto el problema siempre que el fragmento de código funcione cambiando el valor de `x_value`. Si no funciona, se otorga **1 pto.** si es que se entrega la explicación correcta del error dentro de la respuesta.

Pregunta 3: Elabore el código (3 ptos.)

Elabore, utilizando el Assembly del computador básico visto en clases, un programa que a partir de un arreglo de números **positivos** `arr` de largo `len`, obtenga la moda, el mínimo y máximo de este; almacenando el resultado en variables `mode`, `min` y `max` respectivamente. A continuación, se muestra un fragmento de código de ejemplo para el *input* del programa:

```
DATA:
  arr 30
      11
      7
      21
      1
      27
      30
      3
   len 8
   mode -1
   min  127
   max  0
CODE:
  ;Complete aquí su código
```

Si existe más de una moda, un mínimo o un máximo, puede escoger arbitrariamente el valor a almacenar como resultado. Puede agregar todas las variables adicionales que necesite para resolver el problema.

Se otorgan **0.75 ptos.** por encontrar correctamente el mínimo; **0.75 ptos.** por encontrar correctamente el máximo; y **1.5 ptos.** por encontrar correctamente la moda. Por cada valor, se descuenta la mitad del puntaje si su programa falla como máximo en un caso; y no se otorga puntaje si falla en más de un caso.

Solución: El programa puede incluir variables nuevas y recorrer el arreglo las veces necesarias para solicitar lo pedido. Lo importante es que al finalizar la ejecución, las variables `mode`, `min` y `max` estén correctamente actualizadas. El puntaje se asigna tal como se detalla en el enunciado. En la siguiente página, se incluye un programa de ejemplo que cumple lo pedido en este problema.

```
DATA:
  arr 30
      11
      7
      21
      1
      27
      30
      3
   len 8
   mode -1
   min 127
   max 0
   ; Variables auxiliares
   mode_counter 0
   current_value -1
   current_counter 0
   mode_index 0
   index 0

CODE:
  ; Parte 1 - mínimo y máximo
  MOV A,(arr)
  min_max_loop:
    CMP A,(min)
    JLT set_min
    CMP A,(max)
    JGT set_max
    continue_min_max_loop:
      INC (index)
      MOV A,(len)
      CMP A,(index)
      JLE mode_start
      MOV B,(index)
      MOV A,arr
      ADD B,A
      MOV A,(B)
      JMP min_max_loop
    set_min:
      MOV (min),A
      CMP A,(max)
      JGT set_max
      JMP continue_min_max_loop
    set_max:
      MOV (max),A
      JMP continue_min_max_loop



  ; Siguiente página -> Moda
  ; Siguiente página -> Moda
  ; Siguiente página -> Moda
```

```
; Parte 2 - Moda
mode_start:
  MOV A,0
  MOV (index),A
mode_loop:
  MOV B,(index)
  MOV A,arr
  ADD B,A
  MOV A,(B)
  CMP A,(mode)
  JEQ continue_mode_loop
  MOV (current_value),A
  mode_inner_loop:
    MOV B,(mode_index)
    MOV A,arr
    ADD B,A
    MOV A,(B)
    CMP A,(current_value)
    JEQ inc_current_counter
    JMP continue_mode_inner_loop
    inc_current_counter:
      INC (current_counter)
    continue_mode_inner_loop:
      INC (mode_index)
      MOV A,(mode_index)
      CMP A,(len)
      JLT mode_inner_loop
      JMP check_counter
  check_counter:
    MOV A,(current_counter)
    CMP A,(mode_counter)
    JGT change_mode
    JMP continue_mode_loop
  change_mode:
    MOV (mode_counter),A
    MOV A,(current_value)
    MOV (mode),A
  continue_mode_loop:
    MOV A,0
    MOV (current_counter),A
    MOV (mode_index),A
    INC (index)
    MOV A,(index)
    CMP A,(len)
    JGE end
    JMP mode_loop
end:
```