

Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Ciudad de México

- Tarea 3 Programación en Ensamblador -

Modelación de Sistemas Mínimos y Arquitecturas Computacionales *Grupo 233*

Profesores:

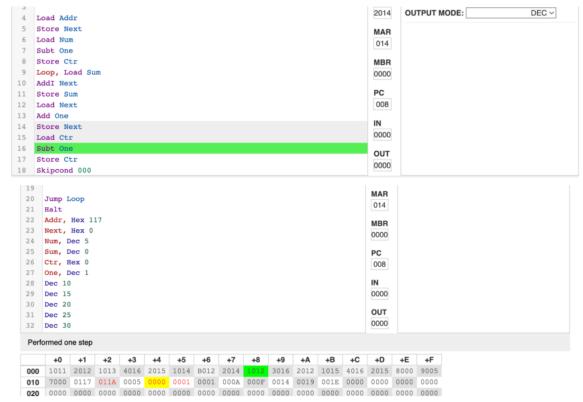
Dr. Yoel Ledo

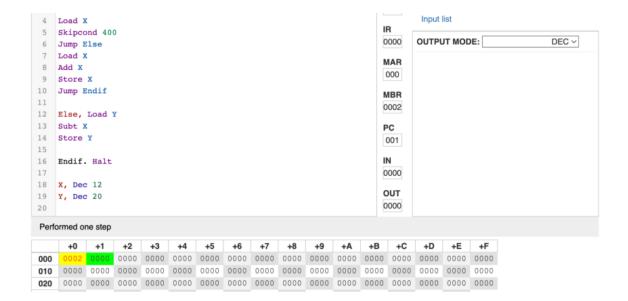
José Luis Almeida Esparza - A01028493

13 de octubre de 2023

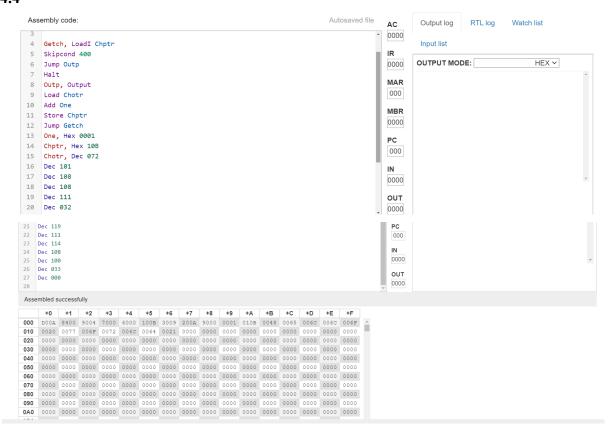
2.-Ensambla y ejecuta los ejemplos 4.2 al 4.4 del libro de texto.

4.2





4.4



Redacta brevemente la respuesta a los conceptos del capítulo 4: "Review of essential terms and concepts".

Reformulación de conceptos y términos clave:

- Función de la CPU: La CPU, conocida como la Unidad Central de Procesamiento, desempeña el papel principal en el funcionamiento de un sistema informático, actuando como su cerebro. Su tarea principal consiste en ejecutar operaciones matemáticas, lógicas y de control de datos, supervisando la operación global de la computadora.
- 2. Propósito del Datapath (Ruta de Datos): El Datapath se utiliza como el núcleo de control para supervisar los procesos. Contiene unidades funcionales, como las unidades de cálculo aritmético y lógico, y registra el procesamiento de datos.
- 3. Función de la Unidad de Control: La Unidad de Control dirige las actividades de la CPU al interpretar y gestionar las operaciones e instrucciones emitidas por otras unidades. Además, se encarga de garantizar la correcta manipulación y procesamiento de los datos.
- 4. Registros y Tipos: Los registros son áreas de almacenamiento de alta velocidad ubicadas dentro de la CPU, que incluyen registros de datos, registros de direcciones y registros de control.
- 5. Determinación de Funciones de la ALU: La Unidad Aritmética y Lógica (ALU) realiza operaciones matemáticas y lógicas, interpretando las operaciones a llevar a cabo según las instrucciones codificadas proporcionadas por la Unidad de Control.
- 6. Cuellos de Botella del Bus: El Bus, que es el sistema de comunicación a través del cual fluye la información, puede volverse ineficiente y lento cuando se incrementa la cantidad y la velocidad de procesamiento.
- 7. Diferencia entre Bus Punto a Punto y Bus Multipunto:
 - Bus Punto a Punto: Conecta directamente componentes específicos.
 - Bus Multipunto: Permite la conexión de múltiples componentes, facilitando la comunicación compartida entre ellos.
- 8. Importancia del Protocolo del Bus: El Protocolo del Bus establece la forma de comunicación y transferencia de información entre los componentes, garantizando la precisión y eficiencia de la transmisión.
- 9. Diferencias entre Buses de Datos, Direcciones y Control:

Bus de Datos: Transfiere información entre los componentes.

Bus de Direcciones: Lleva consigo direcciones de memoria.

Bus de Control: Gestiona señales de control, como las de lectura y escritura.

- 10. Ciclo del Bus: Un ciclo del bus ocurre cuando los datos se transfieren de un componente a través del bus. Este ciclo involucra tres etapas clave: la Fase de Dirección (donde se establecen las direcciones de transmisión), la Fase de Datos (donde se envían datos a través del bus) y la Fase de Control (que determina el tipo de operación que se llevará a cabo). Estos ciclos son cruciales para sincronizar las transferencias de datos en un sistema informático.
- 4.- Realiza un programa que nos diga la longitud de un string, del cual conoces la dirección donde inicia y termina cuando encuentra el NULL en el contenido de memoria; asume que el string está en ASCII

```
1 //Jose Luis Almeida Esparza A01028493
    //set string start adress
   Input
   Store stringStartAdress
   //Ask for ascci chars until input null (0)
   inputLoop, Input
              Skipcond 400
                Jump storeChar //if input != 0
               //else
Jump getLength
13
14
15
16
17
    storeChar, StoreI tempAdress //store input(last AC) into the adress that tempAdress points to
20
21
22
                //increment tempAdress by {\bf 1}
                Load tempAdress
23
               Add one
               Store tempAdress
25
                Jump inputLoop
26
27
28
29 getLength, Load stringStartAdress
                Store tempAdress
32
               Jump stringLoop //if string start adress ptr value !=0
33
34
35
               //else
               Jump haltProgram
36
37
39
    stringLoop, Load stringLength //add one to stringlength
40
                Add one
41
               Store stringLength
42
43
              //increment temp adress by one
Load tempAdress
44
45
46
               Add one
47
               Store tempAdress
48
49
              //if tempadress ptr val == 0 halt else go back to stringLoop
51
               LoadI tempAdress
52
               Skipcond 400
53
54
               Jump stringLoop
55
58
   //halt and print length
59 haltProgram, Load stringLength
60
                   Output
61
                   Halt
62
63
64
```

5.-Realiza un programa que calcule: Y=X-2*A-3*B-4*C

```
1 //Jose Luis Almeida Esparza A01028493
 2
3 //Realiza un programa que calcule: Y=X-2*A-3*B-4*C (programa5.mas).
4
6 // Add x
7 Clear
8 Add X
9
10 //add -2*A
11 Subt A
12 Subt A
13
14 // add -3*B
15 Subt B
16 Subt B
17 Subt B
18
19 // add -4*c
20 Subt C
21 Subt C
22 Subt C
23 Subt C
24
25 Store Y
26 Output
27 Halt
28
29
30 Y, DEC 0
31 X, DEC 0
32 A, DEC 0
33 B, DEC 0
34 C, DEC 0
```

6.-Realiza un programa que calcule: Y=5*(A-B+C)

```
C
1 //Jose Luis Almeida Esparza A01028493
                                                                                    I
3 //Realiza un programa que calcule: Y=5*(A-B+C) (programa6.mas).
                                                                                     C
5
                                                                                    N
6
7 // temp = (A-B+C)
8 Clear
                                                                                    P
9
   Add A
                                                                                     (
10 Subt B
                                                                                    F
11 Add C
12
13 Store TEMP
                                                                                    I
14
                                                                                     C
15
16 //add 5*temp
                                                                                    (
17 Clear
                                                                                    (
18 Add TEMP
19 Add TEMP
20 Add TEMP
21 Add TEMP
22 Add TEMP
23
24
25 Store Y
26 Output
27 Halt
28
29
30 Y, DEC 0
31 TEMP, DEC 0
32 A, DEC 0
33 B, DEC 0
34 C, DEC 0
35
```

7-Realiza un programa que invierta 4 localidades de memoria.

```
0
1
    //Jose Luis Almeida Esparza A01028493
2
                                                                                    IF
3
    //Problema 7. Realiza un programa que invierta 4 localidades de memoria.
                                                                                    0
4
5
        Load A
                                                                                    M
6
        Store T1
                                                                                    (
7
        Load B
8
                                                                                    M
9
        Store T2
                                                                                    0
10
                                                                                    Ρ
11
        Load C
                                                                                    (
12
        Store T3
13
                                                                                    11
14
        Load D
                                                                                    0
        Store A
15
16
                                                                                    0
17
        Load T3
                                                                                    0
        Store D
18
19
        Load T2
20
        Store C
21
        Load T1
22
23
        Store B
24
25
        Halt
26
27
   A, DEC 20
28
   B, DEC 10
   C, DEC 40
29
   D, DEC 30
    T1, DEC 0
31
32
    T2, DEC 0
   T3, DEC 0
33
34
```

8.-Realiza un programa que tome la primera localidad de memoria de una secuencia de 6 números, llenando las restantes cinco con valor de la anterior más uno.

```
//Jose Luis Almeida Esparza A01028493
1
2
   /Programa 8. Realiza un programa que tome la primera localidad de memoria de una secuencia
3
4
   /de 6 números,llenando las restantes cinco con en valor de la anterior más uno
5
6
        Load
              Num
7
       Store 200
8
9
       Load 200
10
       ADD 1
11
       Store 201
12
       Load 201
13
14
       Add 1
        Store 202
15
16
        Load 202
17
        Add 1
18
        Store 203
19
20
        Load 203
21
22
        Add 1
23
        Store 204
24
       Load 204
25
26
        Add 1
        Store 205
27
       Halt
28
29 Num, DEC 2
30
```

9.-realiza un programa que tome una serie de cinco localidades de memoria y a cada una sume un valor: La primera 1, la segunda 2, etc.

```
//Jose Luis Almeida Esparza A01028493
   //load arr start adress into temp adress
Load arrStartAdress
   Store tempAdress
7 //make i the arr size
8 Load i
9 Add arrSize
10 Store i
11
12
               //ask for memory adresses 5 times
   inputLoop, Input
14
               StoreI tempAdress
15
16
               //increment temp adress by one
17
               Load tempAdress
18
               Add ONE
               Store tempAdress
19
20
21
               //decrement i by one
22
               Load i
23
                Subt ONE
24
                Store i
25
26
27
               Skipcond 400
               Jump inputLoop //if i != 0 loop again
28
29
30
   addLoop, Load i//inctement i by 1
31
32
33
               Store i
34
35
36
               LoadI arrStartAdress //load the value that arrStartAdress points to (its another adress/pointer)
                          // add i to the loaded adress
// subt 1 to ofset
37
                Add 1
                Subt ONE
38
39
                Store tempAdress //store the incremented adress
40
41
42
                LoadI tempAdress //load the value that tempadress point to
                Add i //add i to the adress it points to
43
44
                StoreI tempAdress //store the value in the pointed adress
45
46
47
               Load arrSize //load array size Subt i
48
49
                Skipcond 400
                Jump addLoop //if (arrsize - i) != 0 loop again
50
51
52
53 Halt
54
55
56
57
58
   i, DEC 0
59
   ONE, DEC 1
60
   arrSize, DEC 5
61
   arrStartAdress, DEC 128
62
   tempAdress, DEC 128
63
64
```