****

**UNIVERSIDAD TECNOLÓGICA NACIONAL**

**Facultad Regional Buenos Aires**

***ARQUITECTURA de COMPUTADORES [08-2022]***

**–2023–**

**Docente: Prof. Roberto Tenuta**

**Trabajo Práctico grupal Assembler Debug**

***«trabajo práctico grupal Assembler Debug »***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Localización [MEDRANO]** | | | | | | | **Curso: K1029** | | | | | | |
| **Integrantes del equipo: Herzkovich Agustín, Punta Máximo, Palazzesi Tomás, Adorno Elías, Stamati Gad** | | | | | | | | | | | | | |
| **Legajo: 213.787-2** | **Herzkovich Agustín** | | | | | | | | | | | | |
| **Legajo: 214.033-0** | **Punta Máximo** | | | | | | | | | | | | |
| **Legajo: 214.166-8** | **Stamati Gad** | | | | | | | | | | | | |
| **Legajo: 213.961-3** | **Palazzesi Tomás** | | | | | | | | | | | | |
| **Legajo: 213.445-7** | **Adorno Elías** | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | |
| **Entrega / Revisión** | **1** | | | | **2** | | | | | **3** | | | |
| **Fecha de entrega** |  | | | |  | | | | |  | | | |
| **Fecha de calificación** |  | | | |  | | | | |  | | | |
| **Calificación** | **A** | **B** | **C** | **D** | **A** | **B** | | **C** | **D** | **A** | **B** | **C** | **D** |
| **Firma del Docente** |  | | | |  | | | | |  | | | |
| **Observaciones:** | | | | | | | | | | | | | |

**Práctico 1**

1) En las 3 instrucciones se utiliza un modo de direccionamiento inmediato, ya que los operandos se encuentran en el campo dato.

2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instrucción** | **Código de Máquina** | **Código de Operación** | **Campo de Referencia al DATA** | |
| MOV AH,08 | B408 | B4 | 08 |
| ADD AH,03 | 80C403 | 80C4 | 03 |
| SUB AH,04 | 80EC04 | 80EC | 04 |

3) La base es la dirección de inicio de un segmento de memoria. El desplazamiento es la distancia o el número de bytes entre una dirección específica dentro de un segmento y la dirección base de ese segmento.

4) 0CD7:0105.

5) En el caso de este ejemplo con r se modifica el CS, seteándolo en 0CD7. En pasos posteriores simplemente se utiliza para ver el valor de los registros.

6) AX: es un registro general de 16 bits y se utiliza principalmente para operaciones aritméticas.

BX: es otro registro general de 16 bits y se utiliza para almacenar datos generales.

CX: es un registro general de 16 bits y se utiliza en bucles y contadores.

DX: es otro registro general de 16 bits y se usa para diversas operaciones, como entrada y salida.

SP: es el puntero de pila, que apunta a la cima de la pila en la memoria.

BP: es el puntero de base y se usa en funciones y procedimientos para acceder a los parámetros y variables locales.

SI: es el índice de origen, que se utiliza en operaciones de cadena.

DI: es el índice de destino, que también se usa en operaciones de cadena.

DS, ES, SS, CS: son segmentos de datos, segmentos de destino, segmentos de pila y segmentos de código, respectivamente, y se utilizan para direccionamiento en el modo real.

IP: es el puntero de instrucción y apunta a la dirección de la próxima instrucción que se ejecutará.

7)

* NV: No hay overflow.
* UP: Bandera de dirección hacia adelante.
* EI: Interrupciones activadas.
* PL: Signo positivo.
* NZ: No es cero.
* NA: No hay acarreo auxiliar.
* PO: Paridad non.
* NC: No hay acarreo.

8) En el primer trace:

La dirección es 0CD7:0102, donde 0CD7 hace referencia al valor del CS y 0102 hace referencia al valor del IP.

El código de máquina es 80C403, donde 80C4 hace referencia al código de operación y 03 hace referencia al campo data.

La instrucción es ADD AH,03, donde ADD hace referencia a la instrucción de Assembler de suma, AH hace referencia al destino y 03 hace referencia al origen.

**Práctico 2**

1) AH = 11111111b.

El signo lo representa el bit más significativo, y la magnitud los otros 7 bits.

2) -128 a 127 en decimal.

3) 11111111b como entero signado, pasado a complemento a 2: -(00000001b) = -1d

4) No se modificó ninguna bandera.

5)

|  |  |  |  |
| --- | --- | --- | --- |
| **Instrucción** | **Base** | **Desplazamiento(inst)** | **Contenido(registro)** |
| MOV AH,08 | 0CD7 | 0101  0100 | 08  00 |
| ADD AH,03 | 0CD7 | 0104  0103  0102 | 0B  00 |
| SUB AH,0C | 0CD7 | 0107  0106  0105 | FF  00 |

6) La diferencia es que en este caso se carga el valor en todo el registro AX (al ser 8, queda alojado en la parte baja), en los anteriores ejemplos, lo alojábamos en la parte alta, es decir, AH. El código de operación se modifica quedando de la siguiente forma: B8. El cambio de registro afecta al valor del código de operación porque aquí se están modificando los 16 bits del registro, en la versión anterior solo se modificaban los 8 bits más significativos.

7) Se refiere a que lo que está en la posición 0200 va a pasar a valer 0008 debido a la instrucción MOV.

8) E 0200: 08.

E 0201: 00.

Carga dos bytes porque el registro es de 16 bits.

9) Se borró el 0008 y ahora su nuevo contenido es 0003.

El código de operación era B8 y el campo data era 0300.

10) Se modificaron los bytes que estaban en la dirección 0203 con el valor 0003.

11) 0008.

12) 000B.

13) En 0205: 0B.

En 0206: 00.

14)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instrucción** | **Base y Desp** | **Destino** | **Contenido** | |
| MOV AX,0008 | 0CD7:0100 | Registro AH | 00 |
|  |  | Registro AL | 08 |
| MOV [0200],AX | 0CD7:0103 | 0201 | 00 |
|  |  | 0200 | 08 |
| MOV AX,0003 | 0CD7:0106 | Registro AH | 00 |
|  |  | Registro AL | 03 |
| MOV [0203],AX | 0CD7:0109 | 0204 | 00 |
|  |  | 0203 | 03 |
| MOV AX,[0200] | 0CD7:010C | Registro AH | 00 |
|  |  | Registro AL | 08 |
| ADD AX,[0203] | 0CD7:010F | Registro AH | 00 |
|  |  | Registro AL | 0B |
| MOV [0205],AX | 0CD7:0113 | 0206 | 00 |
|  |  | 0205 | 0B |

**Práctico 4**

1) La respuesta es que en AH=01 Tengo el resto de la división (1 decimal) y en AL=02 tengo el Cociente (2 decimal).

2) Dividendo = 28d.

Divisor = -1d.

Cociente = -28d.

**Práctico 6**

1) La funcion del registro BX es la de utilizarse para hacer un loop ya que este se va decrementando, y se itera ya que el salto condicional jnz salta siempre y cuando BX no valga 0.

2) Cuando BX = 4, salta a la posición 138, es decir a la instrucción almacenada en 0138 cambiando el ip a 0138. Cuando BX = 0, jnz no se ejecuta y pasa la siguiente instrucción, quedano el ip igual a 013D.

3) También se pueden llamar interrupciones.

4) Sí.

5)

* JMP (Jump): La instrucción JMP se utiliza para realizar un salto incondicional a una dirección de memoria específica. Cambia el flujo de ejecución sin ninguna condición.
* JNZ (Jump if Not Zero): La instrucción JNZ se utiliza para realizar un salto condicional si la bandera "Zero" (ZF) no está establecida. Esto significa que salta si el resultado de una operación anterior no fue igual a cero.
* CALL: La instrucción CALL se utiliza para realizar una llamada a una subrutina o procedimiento. Guarda la dirección de retorno en la pila y cambia el flujo de ejecución a la dirección de destino.
* RET (Return): La instrucción RET se utiliza para regresar de una subrutina. Recupera la dirección de retorno de la pila y cambia el flujo de ejecución de nuevo al punto de llamada.
* INT (Interrupt): La instrucción INT se utiliza para generar una interrupción, solicitando servicios al sistema operativo o manejando eventos externos.
* IRET (Interrupt Return): La instrucción IRET se utiliza para regresar de una rutina de interrupción. Restaura el estado previo a la interrupción y reanuda la ejecución normal del programa.
* LOOP: La instrucción LOOP se utiliza para implementar bucles. Disminuye un contador y salta a una dirección de destino si el contador no es igual a cero.

6) La interrupción INT 21 se utiliza para realizar una variedad de funciones relacionadas con la administración de archivos y la entrada/salida (E/S). Su función principal es proporcionar una interfaz de servicio para que los programas de MS-DOS realicen operaciones de E/S y administración de archivos.

**Práctico 7**

1)

|  |  |  |
| --- | --- | --- |
| **Sucesivas Instrucciones de Carga** | **Contenido** | **Actualización del SP** |
| PUSH [1100] | B7 | 14 |
|  | B8 | 13 |
| PUSH AX | D6 | 11 |
|  | D5 | 10 |
| PUSH BX | D4 | 8 |
|  | D3 | 7 |
| PUSH CX | D2 | 5 |
|  | D1 | 4 |

2)

|  |  |  |  |
| --- | --- | --- | --- |
| **Sucesivas Instrucciones de Carga** | **Contenido** | **Actualización del SP** | **Contenido del Registro Destino** |
|  | D1 | 4 |  |
| POP AX | D2 | 5 | AX = D1D2 |
|  | D3 | 7 |  |
| POP BX | D4 | 8 | BX = D3D4 |
|  | D5 | 10 |  |
| POP CX | D6 | 11 | CX = D5D6 |
|  | B8 | 13 |  |
| POP DX | B7 | 14 | DX = B8B7 |