Bienvenido: Ingresar
location: WebHome / TrabajosPracticos / PracticoASM3

Trabajo Práctico Nro 3 Assembler x86

Tabla con Instrucciones	IntelCodeTable_es.pdf

Este práctico, tiene como objetivo ejercitarse en la programación del assembler del x86, como así también conocer las herramientas necesarias para resolver las diferentes instancias de un proyecto (Escritura del código, compilación, enlace y debug). Los trabajos se desarrollaran en Linux necesitando solamente un editor de texto plano para la escritura de los prácticos y el paquete de herramientas GNU toolchain generalmente ya instalado en cualquiera de las distribuciones del S.O.

Es conveniente además para la realización de los prácticos en esta primera etapa, utilizar el mismo nombre del archivo fuente y el de los archivos en binario, distinguiendo a cada ejemplo o ejercicio por el nombre del directorio que utilicemos.

Tendremos entonces dentro del directorio de cada ejercicio los siguientes archivo:

Archivo de entrada

- **ex1.s** archivo en texto del programa en assembler

Este archivo está constituido por un encabezado que es fijo para todos los ejemplos

```
.intel_syntax noprefix
.file "ex1.s" # opcional
# seccion de variables
.section .data

# seccion de programa
 .section .text
 .global _start
 _start:

......
......

 mov eax,1 # codigo de salida para al SO
 mov ebx,0 # retorno 0
 int 0x80 # llamada al SO
 .end
```

Archivos de salida

- **ex1.o** archivo objeto
- **ex1** archivo ejecutable

Compilar y Enlazar

Para compilar deberemos utilizar el compilador de assembler del proyecto GNU denominado **GAS** provisto por las binutils, al mismo se accede por el comando **as**

```
 as --gstabs ex1.s -o ex1.o
```

En este caso generaremos un archivo objeto denominado **ex1.o**

Luego para enlazar usaremos el **ld** de la forma

```
ld ex1.o -o ex1
```

finalmente, creamos un archivo **ex1**, el cual será nuestro proyecto ya enlazado y listo para correr

Debug y simulación

Al terminar de compilar un proyecto y al no disponer de ninguna salida en los primeros ejercicios, recurrimos al gdb como herramienta para ver como funciona el programa recién creado.

La herramienta gdb puede correr sola invocando directamente

```
gdb ex1
```

o por medio de una interfaz gráfica, para esto instalamos la aplicación **kdbg** y luego llamando a esta herramienta como

```
kdbg ex1
```

Solución a algunos problemas del kdbg

Ejercicios

Ejercicio 1

Sumar 2 nro de 32 bits contenidos en AX BX y CX DX respectivamente, guardando el resultado en AX BX. AX BX = 0134A23Bh CX DX = BD02E329h

Resultado

AX BX = BE378564 (Resultado)

archivo fuente ex1.s

Ejercicio 2

Usando instrucciones Shift ingresar un número en AX y multiplicarlo por 7

archivo fuente ex1.s

Ejercicio 3

Salvar al los registros AX y BX, luego borrar los mismos para finalmente recuperar los datos del Stack

archivo fuente ex1.s

Ejercicio 4

Dividir AX por 7 y Multiplicar BX por 23

archivo fuente ex1.s

Ejercicio 5

Realizar un programa que: dado un vector de byte ya cargado, busque el final del mismo (byte = 0h) y termine dejando en EBX la longitud del mismo.

archivo fuente ex1.s

Ejercicio 5

Realizar un programa que: dado un vector de byte ya cargado de 10 elementos, sume los mismos y termine con el resultado de la suma en AX.

archivo fuente ex1.s

Práctico de Aula no Desarrollados

Ejercicio 1

Realizar un programa que dada dos variables word previamente cargadas con sendos numero con signo, realice la suma de las mismas y la resta, el tamaño del resultado deben ser tal que no puedan producirse desborde cualquiera sean los números a sumar o restar. (Cuando nos referimos a una cadena o variable previamente cargada, significa que en el momento en que declaran el vector o variable en el programa se ingresan los valores, como en el caso de los ejercicios resueltos 4 y 5)

Ejercicio 2

Realizar un programa que mediante corrimiento divida por 8 un número con signo previamente cargado en una variable word. (Explicar que instrucción uso para el corrimiento y porque)

Ejercicio 3

Usando únicamente las instrucciones de corrimiento y rotación, realice la multiplicación de un número previamente cargado en el acumulador AX por 0Bh, sabiendo que el resultado no será mayor a 16 bits, realice el mismo ejemplo con la función de multiplicación MUL, calcule cuantos ciclos de reloj utiliza cada procedimiento.

Ejercicio 4

Realizar un programa que mediante corrimiento cuente la cantidad de unos que hay en una cadena de 4 bytes previamente cargada, el resultado debe quedar en AL.

Ejercicio 5

Realizar un programa que: Divida por 10 a cada elemento de un vector de Word previamente cargado, el vector no tiene longitud definida, se debe dividir hasta encontrar un 0h, los resultado se guardan en el mismo vector y deberá ser Word para evitar sobreflujo, el resto de la división de descarta.

UntitledWiki: WebHome/TrabajosPracticos/PracticoASM3 (última edición 2012-08-22 18:43:04 efectuada por GuillermoSteiner)