Bienvenido: Ingresar
location: WebHome / TrabajosPracticos / PracticoASM4

Trabajo Práctico Nro. 4 Assembler del ARM

Tabla con Instrucciones	ARM_Instruction_Set.pdf

Este práctico, tiene como objetivo ejercitarse en la programación del assembler del ARM, como así también conocer las herramientas necesarias para resolver las diferentes instancias de un proyecto (Escritura del código, compilación, enlace y simulación).

Para la realización de estos prácticos, deberemos instalar las herramientas GNU toolchain compiladas para ARM, las mismas se pueden bajar en:

HERRAMIENTAS DE PROGRAMACIÓN

Las herramientas provistas para compilar, enlazar, depurar, etc, poseen el mismo nombre que aquellas utilizadas en la arquitectura x86, para distinguirlas se antepone "arm-elf-","arm-elf-eabi-" o "arm-none-eabi-" según la versión del compilador, indicando así que son herramientas compiladas para ARM.

Todos los ejercicios se realizan en base al microcontrolador LPC 2114, pero no se hace referencia al hardware de periféricos, lo que permite la fácil simulación.

Es conveniente 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

```
.text
.arm
.global _start
_start:
         b reset
         b loop
         b loop
         b loop
         b loop
         nop
         b loop
         b loop

 /* ============================
 *  CODIGO
 *  ============================
 */

 reset:
......
```

```
......
```

```
......
......
 loop:   b loop
 /* ============================
 *  CONSTANTES
 *  ============================
 */
 CTE1: .word 0x1231AAAA
 .end
```

Archivos de salida

- **ex1.elf** archivo binario
- **ex1.lst** archivo con el código fuente y el binario desensamblado.

Compilar y Enlazar

Para compilar deberemos utilizar el compilador de assembler provisto por las binutils denominado **as** pero compilado para ARM (compilador cruzado).

```
arm-elf-as -mcpu=arm7tdmi -g ex1.s -o ex1.o
```

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

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

```
arm-elf-ld -Ttext=0 -g ex1.o -o ex1.elf
```

Finalmente, creamos un archivo **ex1.elf**, el cual será nuestro proyecto ya enlazado y listo para simular.

Otra forma mas directa, es utilizando los recursos que nos da el compilador **gcc**, el mismo al recibir un archivo fuente en assembler invoca internamente al compilador **as** luego si se requiere como archivo de salida un binario (directiva -o), internamente llama entonces al enlazador **ld**, de esta forma y utilizando al **gcc** únicamente como herramienta que automatice el compilado en assembler y posterior enlace nos queda.

Linea de comando para compilar y enlazar el archivo

```
 arm-elf-gcc -mcpu=arm7tdmi -g -nostartfiles ex1.s -o ex1.elf
```

Archivo de salida **ex1.elf** archivo binario.

Otros archivos de salida

Listado con el código fuente intercalado con el desensamblado del archivo binario (muestra información sobre posición donde se guarda los datos y programa, valores de los saltos, etc).

```
arm-elf-objdump -S ex1.elf > ex1.lst
```

Archivo de salida **ex1.lst**

Debug y simulación

Al terminar de compilar un proyecto, y ante la imposibilidad de poder correrlo en una PC por ser de otra arquitectura, debemos recurrir a un simulador, en este caso usamos el provisto por las GNU binutils, el cual simula solamente el núcleo del ARM, permitiendo ver como se modifican los registros y la memoria del micro por cada instrucción de la aplicación probada.

gdb --tui

Para realizar un debug con esta herramienta no es necesario instalar nada.

Se debe ejecutar

```
arm-elf-eabi-gdb --tui
```

luego dentro de este programa se escribe.

```
target sim
file ex1.elf
load
```

Aparecerá en una ventana el fuente a depurar, escribiendo. `break nn` Se realiza un break point (nn es el número de linea que te aparece al costado izquierdo del archivo fuente).

Un tutorial de la herramienta se puede ver en

http://sourceware.org/gdb/onlinedocs/gdb/TUI.html

para recordar los comandos mas usados

Comandos Debug

Ejercicios

Ejercicio 1

Realizar un programa que sume dos números de 64 bits cada uno, guardando el resultado en R5 R6 (64bits), los números a sumar estarán dispuesto en R1 R2(64bits) y R3 R4(64bits) respectivamente.

archivo fuente ex1.s

Ejercicio 2

Realizar un programa que analizando un bit en el registro r3 active una alarma si este bit es 0, el nro de bit a analizar está guardado en el registro r4, la alarma se activa escribiendo un 1 en el bit 16 de r6.

archivo fuente ex1.s

archivo fuente ex1.s

Ejercicio 3

Realizar una serie de subrutinas que realizen diferentes comparaciones, devolviendo r0 = 0 si estas comparaciones fueron falsas y r0<>0 si fueron verdaderas, las comparaciones son las siguientes.

- r1<100 y r1>20

- r1<100 o r2>20
- r1 = 10 o r1 = 15 o r1 = 20
- r1 = 10 y r2 = 15 y r3 = 20

archivo fuente ex1.s

Ejercicio 4

Realizar un programa que dado un numero guardado en R1 lo multiplique por 10, de dos formas distintas, usando la instrucción mul y por corrimiento.

archivo fuente ex1.s

Ejercicio 5

Realizar un programa que dado un numero guardado en R1 y otro en R2, calcule el resultado de elevar R1 a la potencia de R2 y guardar el resultado en R3 (R3 = R1^R2).

archivo fuente ex1.s

Ejercicio 6

Realizar un programa que sume los primeros 100 números naturales.

archivo fuente ex1.s

Ejercicio 7

Realizar un programa que dada una cadena con terminación nula guardada en VECT, la pase a mayúscula, guardando el resultado en el mismo vector VECT.

nota:la cadena de entrada solo contendrá valores alfabéticos o espacio en blanco ('a' - 'z','A' - 'Z').

archivo fuente ex1.s

Ejercicio 8

Realizar un programa que dado un vector de 16 halfword con signos, guarde en R1 el resultado de su promedio.

archivo fuente ex1.s

Ejercicio 9

Realizar un programa que sume los word de un vector VECTW sin signo en las posiciones indicadas por un segundo vector VECTB de 20 byte,el resultado guardarlo en R1.

ejemplo

VECTB = 1,3,2,0,2.....

VECTW = 1,111,222,333,444,555,666,.....

se suman 111+333+222+1+222...

archivo fuente ex1.s

Ejercicio 10

Dado un vector de 16 words con signo, realizar un programa que encuentre el elemento del vector mas cercano a la media del mismo.

archivo fuente ex1.s

Ejercicio 11

Dada una cadena de caracteres con terminación nula la cual contiene palabras separadas por espacio, realizar un programa que pase a minúscula todas las letras con excepción de la primera de cada palabra que deberá ser pasada a mayúscula, además, las palabras de menos de 3 letras deberán quedar en minúscula.

ejemplo

entrada

VECT = "El mUndo dE Hoy",0

salida

VECT = "el Mundo de Hoy",0

archivo fuente ex1.s

Ejercicio 12

Se debe realizar un programa que calcule de los 100 primeros números naturales cuales son primos usando el método de la "Criba de Eratóstenes". El método se resume en estos pasos:

- se carga un vector de 99 elementos con los números naturales del 2 hasta 100.
- se comienza por el numero 2, lo dejamos y eliminamos a partir de allí todos los números múltiplo de 2.
- luego nos corremos hasta el próximo número no borrado (en este caso el 3) y eliminamos todos los múltiplos de 3.
- el siguiente numero sin eliminar será el 5, lo dejamos y eliminamos todos los múltiplos de 5.
- así vamos avanzando, cuando llegamos a un número que no ha sido eliminado lo dejamos y a partir de allí eliminamos los múltiplos de él.

Al final copiamos a un nuevo vector todos los números no eliminados.

archivo fuente ex1.s

Ejercicio 13

Se debe realizar un programa que invierta la posición de las letras en cada una de las palabras guardadas en un vector. Las características de este vector a tener en cuenta son:

- Las palabras en el mismo están separadas por uno o varios espacios.
- No existen signos de puntuación.
- El vector termina con nulo.

ejemplo

vec1 = " Las palabras en el mismo están separadas por uno o varios espacios ",0

resultado

vec1 = " saL sarbalap ne le omsim nátse sadarapes rop onu o soirav soicapse ",0

archivo fuente ex1.s

Ejercicio 14

Partiendo de una cadena de caracteres y 10 vectores de 10 punteros cada uno, se pide realizar

un programa que guarde en esos vectores las palabras encontradas en la cadena de la siguiente forma: En el primer vector todas las palabras que tengan menos de 3 letras, luego en el segundo las que tengan 4 letras, en el próximo 5, hasta llegar al décimo vector, donde se guardarán aquellas que tengan mas de 11 letras. En caso de poseer mas de 10 palabras de una clase, no se guardarán.

Nota: no es necesario guardar las palabras en otro vector, simplemente con guardar el puntero de comienzo de la palabra y luego cambiar el espacio que separa esta de la próxima por Nulo es suficiente.

archivo fuente ex1.s

Ejercicio 15

Dada una cadena de bytes sin signo de terminación nula, por cada elemento de esta cadena se sacará el promedio del mismo con los tres bytes anteriores guardando este valor en una segunda cadena en la posición correspondiente. La excepción a esta regla son los tres primeros valores que son guardados directamente en la segunda cadena sin calculo de promedio.

Ejemplo

cadena1 = 2,14,7,8,9,2,.....,0

salida

cadena2 = 2,14,7,7,9,6

donde 2 <- 2, 14 <- 14, 7 <- 7, 7 <- prom(2+14+7+8), 9 <- prom(14+7+8+9), 6 <- prom(7+8+9+2)

archivo fuente ex1.s

Ejercicio 16

Una cadena de caracteres debe se encriptada, para ello se utilizará un método muy básico, el cual consiste en sumar al código ascii de cada letra en la cadena a encriptar el código ascii del caracter correspondiente de una clave, cada vez que se alcanza el final de esta, se vuelve a comenzar con su primer caracter.

Ejemplo

aencrip = 'h','o','l','a',' ','m','u','n','d','o',0

clave = 'f','1','2',0

resultado = 'h'+'f' , 'o'+'1' , 'l'+'2' , 'a'+'f' , ' '+'1' , 'm'+'2' , 'u'+'f' , 'n'+'1' , 'd'+'2' , 'o'+'f'

archivo fuente ex1.s

Ejercicio 17

En un vector de caracteres, el cual posee solo palabras separadas por espacios, se debe buscar la palabra mas larga, luego, formar otro vector con las mismas palabras con el agregado a cada una de n caracteres '_' donde n es la diferencia entre la palabra mas larga y la palabra actual.

ejemplo

```
entrada = "igualar el largo de las palabras",0

salida = "igualar_ el______ largo___ de______ las_____ palabras",0
```

archivo fuente ex1.s

Ejercicio 18

Realizar un programa que elimine de un vector de byte sin signo previamente cargado, los números múltiplos de 4.

El vector se denomina vec1 y termina con 0.

ejemplo

```
entrada en vec1 = 80,5,6,18,4,5,20,19,0

salida en vec1 = 5,6,18,5,19,0
```

archivo fuente ex1.s

Ejercicio 19

Dado dos vectores previamente cargado de caracteres alfanuméricos (letras mayúsculas y minúsculas del alfabeto ingles, espacios y números) y terminación nula, desarrollar un programa que copie del vector 1 a un nuevo vector 3 donde se insertará por cada caracter que coincida con el primer caracter del vector 2 (COMPARACIÓN CASE INSENSITIVE) el propio vector 2 (sin su primer caracter).

Ejemplo

```
Entrada
vector1 = "insertar en 1 la palabra 1"
vector2 = "1mundo"
Salida
vector3 = "insertar en mundo la palabra mundo"
```

archivo fuente ex1.s

Ejercicio 20

Dado dos vectores previamente cargados de caracteres alfanuméricos (letras mayúsculas y minúsculas del alfabeto ingles, espacios y números) y terminación nula, desarrollar un programa que cuente la cantidad de veces que el vector 2 se repite en el vector 1.

Ejemplo

```
Entrada
vector1 = "esto es una prueba de palabras repetidas"
vector2 = "es"
Salida
cantidad = 2
```

archivo fuente ex1.s

Ejercicio 21

Dado un vector previamente cargado de caracteres alfanuméricos (letras mayúsculas y minúsculas del alfabeto ingles, espacios y números) y terminación nula, desarrollar un programa que por cada carácter verifique cuantos caracteres a continuación están repetidos, se dispone para guardar los datos de dos vectores, uno donde se dispone el carácter (tipo byte) y en otro la cantidad de repeticiones de ese carácter (tipo halfword).

Ejemplo

```
Entrada
vector = "fssdffjllllleiiiilv4"
Salida
letras =  'f','s','d','f','j','l','e','i','l','v','4'
cantidad = 1 , 2 , 1 , 2 , 1 , 4 , 1 , 4 , 1 , 1 , 1
```

archivo fuente ex1.s

Ejercicio 22

Realizar una función que reciba como parámetro el puntero a una cadena de terminación nula, la misma deberá transformar la cadena recibida a un número.

Finalmente se deberá utilizar la función en un bucle donde: se tome un valor de un vector que posee el puntero a la cadena (se supone ya cargado), se envíe a la función y el resultado se guarde en otro vector (no es necesario inicializarlo), la acción se repite hasta concluir los 16 ciclos de conversión.

IMPORTANTE:

- El número en ASCII que se encuentra en la cadena es base 16.
- La cadena recibida se supone que solo posee dígitos hexadecimales -donde las letras se encuentran en mayúsculas-, puntos de separación de miles que deben ser descartados o un signo.
- La cadena nunca contendrá un número mayor a 32 bits.
- El único signo posible es un '-' y como primer byte, indicando que el numero es negativo, si este signo no existe el número es positivo.

archivo fuente ex1.s versión 1 archivo fuente ex1.s versión 2

Ejercicio 23

Partiendo de un vector previamente cargado con una cadena alfanumérica de terminación nula, se debe construir un programa que encuentre los números (base 16) contenidos en la cadena, estos números serán cargados en dos vectores de la siguiente forma: en el primero la posición del primer dígito dentro de la cadena (o su puntero), en el segundo vector la longitud de dicho número, esto se repetirá hasta terminar de barrer la cadena.

IMPORTANTE:

- Las características de los números a encontrar serán las siguientes: comienzan con un dígito decimal (0 a 9) o el signo menos (-), pueden contener separaciones de miles (.) o coma decimal (,)
- Se supone los vectores destino ya inicializados y con capacidad suficiente (ambas de tipo word).

Ejemplo

```
entrada = "hola 98A,Bmundo-0A.334"
posicion = 5,15
largo = 5,7
```

archivo fuente ex1.s

UntitledWiki: WebHome/TrabajosPracticos/PracticoASM4 (última edición 2015-07-28 19:35:15 efectuada por GuillermoSteiner)