Universidad Central de Venezuela

Escuela de Computación

Organización y Estructura del Computador I

Proyecto 01

Nombres: Ruben Maza

CI: 21.534.450

Caracas, marzo del 2017

Introducción

Se plantea una calculadora que resuelve sumas, restas, and, or y xor binarios e hexadecimales, via un Shell donde se introducen las operaciones, además de instrucciones adicionales como setsys que cambia el modo de entrada (que puede ser binaria o hexadecimal), chsys que permite el cambio de este mismo.

Debido a cuestiones de tiempo ya que trabajo y estudio e hice el proyecto solo, este proyecto solo hace AND, OR y XOR binarios; la suma y resta está parcialmente programada, reconoce todos los strings e imprime resultados binarios.

Se utilizaron conceptos de divisiones sucesivas para introducir los valores a los arreglos, al igual que comparaciones sucesivas entre strings establecidos y saltos y procedimientos. Entre otros.

Diseño e implementación de la solución

En .section .data

Introducimos todos los string de comparaciones, esto para poder hacer el "shell", como se puede ver:

```
shell:      .asciz ">>"
string:     .asciz "%s"
```

….

Ademas colocamos strings que utilizaremos para impresion:

```
separa1:    .asciz "+ "
separa2:    .asciz "  "
salto:      .asciz "\n"
```

Los Valores y bancos de memoria temporales que utilizaremos a lo largo

```
valor1:         .long 0
valor2:         .long 0
```

...

Luego los, arreglos de 8 para binario, ya que nuestras operaciones seran de 1 byte

```
arreglo1:   .long 0,0,0,0,0,0,0,0
arreglo2:   .long 0,0,0,0,0,0,0,0
resulbin:   .long 0,0,0,0,0,0,0,0
```

E igualmente para los hexadecimales

```
par1:     .long 0,0
par2:     .long 0,0
resulhexa: .long 0,0
```

Algunas banderas para ayudarnos con los cambios de base

```
#flags
setsysflag: .long 0   #si es 2 es que esta en binario, si esta en 16 es que es hexadecimal, si esta
chsysflag:  .long 0
```

Y finamente, mensajes del shell y de errores

```
#mensajes shell#
base2:              .asciz "Cambiando a sistema en base 2 \n"
base16:             .asciz "Cambiando a sistema en base 16 \n"
...

#mensajes error#
err_chsys:  .asciz "Error de CHSYS, no se ha establecido SETSYS \n"
 ...
```

En: .section .bss

Solo tenemos 1 variable no constante que utilizaremos para que guarde temporalmente,

```
temp:          .space 4
```

Inicializamos con el programa

.globl _start

_start:

inicializacion: **<- Aqui iniciamos los valores de las variables globales que utilizaremos**

mensaje_shell: **<-- Aqui imprimimos el mensaje ">>" del shell"**

leer: **<---leemos el string proporcionado por el usuario**

Deteccion de strings
##

#

Aqui, detectamos con ayuda de las variables constantes string, que instruccion está introduciendo el usuario, y redirigimos segun la opcion a sus segundas verificaciones de numeros

exits: #validamos si es exit

chsyss: #validamos si es chsys, solo se puede usar cuando el flag de chsys esta desactivado (ya se ha usado setsys)

setsysnum: #validamos si es setsys

adds:

subs:

ands:

xors:

ors:

condicional_evitar: #evitamos que el programa siga a los condicionales

jmp mensaje_shell

Verificaciones
##
##

Aqui verificamos chsys y los numeros que se estan poniendo a setsys para hacer el cambio

setnum:

set2:

set16:

chsysverificacion: #verificamos si chsys puede usarse, si no es 1, es q no se ha usado setsys por primera vez

chsyscambio:

procedimientos separacion binario y hexa
####################################

En este módulo, procedemos a verificar en que base estamos, y verificamos esto, para redirigir lo que escribió el usuario a su procedimiento respectivo, si es binario o hexadecimal, y su tipo de operación

leer_valores: #procedimiento de lectura de valores

add_proc:

sub_proc:

and_proc:

or_proc:

xor_proc:

procedimientos binario
##

Aqui primeramente transformamos los "enteros" a "arreglos", para poder hacer las operaciones, add y sub estan parcialmente desarrollados

And, or y xor tienen casi el mismo codigo con variaciones, el algoritmo creado hace un recorrido de 0 a 7 en ambos arreglos, va guardando el valor de cada posicion en dos registros, y los mismos se les realiza una operacion logica segun sea el caso: and, or o xor, y su resultado se introduce en el arreglo de resultado

llenar_arreglos:

#add y sub son lo mismo, son sumas

sub_binario:

add_binario:

and_binario:

or_binario:

xor_binario:

procedimientos hexa
##

Estos procedimientos no fueron implementandos por falta de tiempo, pero siguen la misma lógica que la operaciones binarias

#and y sub son lo mismo son sumas

sub_hexadecimal:

add_hexadecimal:

and_hexadecimal:

or_hexadecimal:

xor_hexadecimal:

impresion procedimientos binarios
################################

Procedimientos sonde llenamos el arreglo, haciendo divisiones sucesivas y siguiendo con el resto en cada iteracion, e imprimimos el arreglo con los estatutos descritos en el documento del proyecto con los strings guardados en .data

llenar_arreglo1:

llenar_arreglo2:

impresion_binaria:

impresion procedimientos hexadecimales
##############################

no implementado, pero sigue la logica de la impresion binaria

impresion_hexadecimal:

mensajes
##
########

Mensajes de errores añadidos para mi para control en el shell

mensaje_chsys: #mensaje de error cuando no hemos establecido primeramente setsys antes de usar chsys

mensaje_setnum: #mensaje de error cuando introducimos un sistema invalido, solo puede ser binario o hexa

mensaje_setsysnochsys: #mensaje de error cuando estamos usando un sistema y usamos setsys sin haber usado chsys

mensaje_sistemainvalido: #mensaje de error, cuando se hace una operacion sin seleccionar un sistema 2 o 16

fin
##
#############

Fin del programa

fin: #finaliza programa

```
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Conclusión

Finalmente, no pude completar totalmente el proyecto por falta de tiempo ya que trabajo y estudio, también pienso que se debió dar un poco más de días.

Se afianzaros los conocimientos de las instrucciones de ensamblador.