UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
ORGANIZACION Y ESTRUCTURA DEL COMPUTADOR I

Microproyecto I

Integrantes:
Martínez Jesús
C.I.: 2119440
jueves 1-3pm

Caracas, 20 de Marzo de 2017

Introducción

Como requisito para cumplir con el Microproyecto I de la materia, se pidió realizar una shell para operaciones de números binarios y hexadecimales en el lenguaje ensamblador IA-32, específicamente: suma, resta, and, or y xor.

Usando los conceptos aprendidos en clases referentes a los números binarios en CA2, cambio de base, suma y resta, se implementó el código en dicho lenguaje las operaciones requeridas.

Diseño de la solución

El programa cuenta con 8 acciones:

1- **setsys:** Selecciona la base en la cual se quiere operar.
2- **chsys**: Permite cambiar la base.
3- **add:** Realiza la suma de dos operando.
4- **sub:** Realiza la resta de dos operando.
5- **and:** Realiza la operación lógica “and” bit a bit de los operando.
6- **or:** Realiza la operación lógica “or” bit a bit de los operando.
7- **xor:** Realiza la operación lógica “xor” bit a bit de los operando.
8- **exit**: salir del programa.

Al ejecutar el programa, lo primero que se debe saber es el tipo de base a operar para luego esperar por una acción (add, sub, and, or, xor) seguido de sus operando.

Para lograrlo, se hizo un ciclo principal “Do” que tiene como condición de salida la acción a ejecutar “exit”. En él, se piden la acción y los números a operar, en el caso de ser hexadecimales pasan primero por una función para transformarlos a binarios y luego realizar la operación con ellos.

Al ingresar en una acción, el programa opera bit a bit los dos números (ya leídos previamente), imprimiendo luego los originales con su operando y el resultado para luego volver al shell y esperar por la siguiente acción.

```
Inicio:
leer(menu)
if(menu==add) go to add;
if(menu==sub) go to sub;
if(menu==and) go to and;
if(menu==or) go to or;
if(menu==xor) go to xor;
add:
        cont=7;
        res=0;
        do:
        suma=0;
        if(n1[cont]==1) suma++;
        if(n2[cont]==1) suma++;
        suma=suma+res;
        if(suma==0){ n3[cont]=0; res=0;}
        if(suma==1){ n3[cont]=1; res=0;}
        if(suma==2){ n3[cont]=0; res=1;}
        if(suma==3){ n3[cont]=1; res=1;}
        cont--;
        if(cont>=0) go to do;

        if (mode==2) imprimir(n3);
        else
          if(mode==16) imprimirhex(n3);
```

```
go to inicio

sub:
        complemento(n2); // invierte los bits y suma uno
        go to add:

and:
        cont=0
        do:
        if (n1[cont]==0) n3[cont]=0;
        else
          if (n2[cont]==0) n3[cont]=0;
          else
            n3[cont]=1;
        cont++;
        if (cont<=7) go to do;

        if (mode==2) imprimir(n3);
        else
          if(mode==16) imprimirhex(n3);

        go to inicio;

or:
        cont=0
        do:
        if (n1[cont]==1) n3[cont]=1;
        else
          if (n2[cont]==1) n3[cont]=1;
          else
            n3[cont]=0;
        cont++;
        if (cont<=7) go to do;

        if (mode==2) imprimir(n3);
        else
          if(mode==16) imprimirhex(n3);

        go to inicio;

xor:
        cont=0
        do:
        if (n1[cont]==1) go to s1
          if (n2[cont]==1) n3[cont]=1;
          if (n2[cont]==0) n3[cont]=0;
          go to fxor
        s1:
```

```
   if (n2[cont]==0) n3[cont]=1;
   if (n2[cont]==1) n3[cont]=0;
fxor:
cont++;
if (cont<=7) go to do;

if (mode==2) imprimir(n3);
else
   if(mode==16) imprimirhex(n3);

go to inicio;
```

Implementación de la solución

Basado en el esquema anterior y usando el lenguaje ensamblador IA-32, se puede tomar como ejemplo la acción “and”:

```
AND:
        #imprime la operacion
        leal Sand, %eax #imprime & por pantalla
        pushl %eax
        call printf
        addl $4, %esp

        #efectua la comparacion
        movl $0, %ebx
        nd1:
                #compara digito de n1 con 1
                movb n1(%ebx), %al
                movb uno, %dl
                cmp %dl, %al
                jne falso5

                #compara digito de n2 con 1
                movb n2(%ebx), %al
                movb uno, %dl
                cmp %dl, %al
                jne falso5

                #guarda 1 en respuesta
                movb uno, %al
                movb %al, resultado(%ebx)
                jmp FAND

                #guarda 0 en respuesta
        falso5:
                movb cero, %al
                movb %al, resultado(%ebx)


        FAND:
                incl %ebx
                cmpl $7, %ebx
                jle nd1




        #prueba si el numero es hexadecimal para impirmir
        cmpl $16, mode
```

```
        je andh

        #imprime resultado e binario
        call imprimir_respuesta_bin     #imprime respuesta
        jmp ciclo

andh:
        #imprime respuesta hexadecimal
        call imprimir_respuesta_hexa    #pasa respuesta a hexa para imprimir
        jmp ciclo
```

Conclusión

Con este proyecto se pone a prueba los conocimientos del contenido de primer y segundo parcial. Los conceptos básicos como la suma y la resta de números en CA2, cambio de base y el uso del lenguaje ensamblador IA-32.

Mediante la implementación del código anteriormente nombrado, pude obtener resultados correctos de dichas operaciones dando como exitoso el proyecto.