



Universidad Autónoma de
Baja California

Facultad de Ciencias
Químicas e Ingeniería

Organización de las Computadoras y Lenguaje Ensamblador

Instrucciones del Procesador 8088.

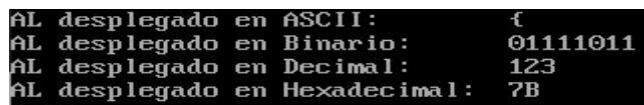
Procedimientos

Teoría

Nuestro sistema de numeración decimal está basado en el concepto más general de sistema numeración posicional. En el sistema posicional el mismo símbolo puede representar diferentes valores dependiendo por su posición en el numeral. En contraste esto es como los símbolos funcionan en un sistema no-posicional, ej. Números Romanos donde X siempre representaría 10. La base de un sistema numérica representa la cantidad de símbolos que contiene. Los sistemas posicionales utilizan exponenciación para determinar el valor del símbolo basada por su lugar. Utilizando este concepto se logra convertir de cualquier base a decimal.

Desarrollo

Se ensamblo, encadeno y ejecuto el programa 'pra9.asm' utilizando atom y su paquete 'atom x86 syntax' se facilitó este labor. El resultado del programa se demuestra en la siguiente imagen.



```
AL desplegado en ASCII: 123
AL desplegado en Binario: 01111011
AL desplegado en Decimal: 123
AL desplegado en Hexadecimal: 7B
```

La implementación del método **printNumBase** utiliza como valor principal a convertir un número decimal, la base a convertir es libre y puede ser seleccionado por el usuario. Una vez teniendo el valor decimal y la base deseada a convertir se almacena en ax el valor decimal, en bl la base, bh se le asigna 00 para asegurar el valor deseado. Se inicializa la etiqueta **initLoop** con la división bl y almacenando su resultado en cx. Una vez modificado el valor cx se almacena el CL en AL ingresamos CH en CL y a CH se le ingresa un 0 así se obtiene el residuo de la división en cx y lo ingresamos en la pila para almacenar el dato. Se ingresa un 0 en AH para tener cociente sin ningún dato extra se incrementa el contador bh para determinar cuántos 'push' se han ejecutado. Procediendo el contador comparamos el cociente almacenado en ax con 0, si no es igual se ejecuta un salto al inicio de la etiqueta caso contrario inicia la etiqueta **impLoop** con un pop almacenando el dato en cx seguido de un decremento de bh utilizando la interrupción 21h con el comando 02h se imprime el contenido de cx, se compara el valor de bh con 0 si no es igual se ejecuta un salto al inicio de la etiqueta caso contrario se termina el proceso y regresa al programa principal.

Conclusión

Los lenguajes de alto nivel contienen diversas funciones que nos permiten elaborar soluciones más complejas con menos código. El limitante que tenemos en el lenguaje ensamblador x86 es la cantidad de registros que utilizamos y el hecho que dependemos de saltos para generar funcionalidades similares alas de if o un ciclo while. Una vez teniendo la

lógica del programa desarrollada es cuestión de decidir la organización del flujo de los datos y en qué registros se almacenaría la información.