Escuela de Computación
Bachillerato en Ingeniería en Computación

TEC | Tecnológico de Costa Rica

Proyecto #1

CalculadorASM

IC-3101 Arquitectura de Computadoras

Tecnológico de Costa Rica, Sede Central Cartago

Escuela de Computación, Ingeniería en Computación

I Semestre 2017

Prof. Ing. Esteban Arias Méndez

NASM es un ensamblador libre, creado para la plataforma de Intel x86. Su objetivo fue la creación de programas de 16 bits, pero se adaptó para que pudiera generar programas de 32 bits, incluso, con las bibliotecas adecuadas, NASM permite la creación de programas portables para cualquier sistema operativo en x86 de 32 bits. En este trabajo se pone en práctica el poder de programación a bajo nivel que provee.

INTRODUCCIÓN

Propósitos

- Poner en práctica los conocimientos aprendidos del lenguaje ensamblador y la utilización de los componentes de NASM.
- Poder aplicar de forma correcta el conjunto de instrucciones del lenguaje ensamblador, para brindar una respuesta adecuada a los problemas.
- Poder mejorar la creación de algoritmos, los cuales le serán útil, para futuros cursos.

Ayuda útil:

Para la utilización del NASM, debe instalar una máquina virtual, acá un enlace donde enseñan cómo hacerlo. (Si aún no lo ha hecho), además de los capítulos 5 al 8 del libro de Sivarama mencionados en clase.

https://www.youtube.com/watch?v=JbJSHnvagFs

Si desea, puede realizar una partición de su disco, donde una parte va a tener algún sistema de Linux, esto es: cada vez que arranque la computadora, podrán elegir con cual sistema operativo iniciar (esto es opcional, si instala la máquina virtual, ya esto no hace falta). Este método de doble boot hace que Linux funcione al 100% del rendimiento de la máquina y por ende la mejor opción.

https://www.youtube.com/watch?v=-UwzcoMWljQ

Para tener referencias con NASM, sobre su instalación, compilación, etc. Haga uso del manual de la herramienta disponible en línea y los capítulos de Sivarama mencionados antes.

http://www.nasm.us/xdoc/2.11.08/html/nasmdoc0.html

PROGRAMACIÓN

El objetivo de este proyecto es la programación en ensamblador NASM con un nivel de complejidad mayor a los ejercicios realizados hasta ahora. Se busca que se utilicen las funciones y conceptos avanzados de NASM como el uso de macros, procedimientos, programación en módulos, pasos de parámetro, el uso de la pila de programa en todas sus posibilidades y la recursividad.

Se trabajará sólo sobre el Sistema Operativo Linux, usando solamente NASM. Se recomienda trabajar su código en módulos de NASM, es decir en varios archivos para facilitar el trabajo. Debe hacer uso del git de la Escuela de Computación para llevar todo el desarrollo de su proyecto.

Este trabajo es para ser desarrollado por parejas de 2 personas máximo.

Según se definió en clase, el trabajo consiste en la realización de una calculadora tipo intérprete, que recibe operaciones aritméticas, brinda el resultado según los detalles siguientes y espera por una nueva expresión a evaluar hasta que el usuario decida terminar el programa

Su calculadora deberá trabajar tomando en cuenta las siguientes consideraciones:

1. Las operaciones básicas a implementar son:
    a. suma, resta, división y multiplicación entre números enteros
    b. conversión de números en punto flotante a binario y viceversa
    c. mostrar la operación complemento a base 2
2. Se debe soportar el ingreso de datos en múltiples bases: binario, octal, hexadecimal y decimal. Para esto el usuario ingresará una cifra numérica e indicará mediante una notación a escoger la base de la cifra: B, O, H, D, b, o, h, d, bin, oct, hex o dec. Dicha notación puede aparecer antes o después de la cifra numérica. La pareja de trabajo puede escoger la notación a usar y el formato en la cifra. Ej.: B###, ###B, bin###, ###bin, b###, ###b, etc. Si no se indica la base el default será decimal.
3. Se deben soportar cifras numéricas representables en 32 bits máximo usando notación de complemento a base 2, es decir desde 2^-31 hasta (2^31)-1. La representación máxima para punto flotante será de 32 bits.

4. Se soportan operaciones con cifras en positivo o negativo. Ej. 4 – 5, +4 – 5, +4 – +5, darán -1 como resultado. Considere casos como 4 – -5 que dará como resultado 9.
5. En una misma expresión se pueden combinar distintos tipos de bases numéricas.
6. Para las conversiones en punto flotante, al aparecer el símbolo '.' en la cifra numérica se entenderá que se desea calcular su representación decimal o binaria según el caso. Si la base ingresada para convertir a binario es diferente a la decimal, se deberá convertir la cifra a decimal y luego calcular el valor en binario.
7. Para calcular el complemento a base 2 de un número puede usar el símbolo '~'. Ej. ~ 1234oct, convertirá la cifra 1234 octal a su valor binario correspondiente y brindará el complemento de dicha cifra. Se deben mostrar la cifra en binario y el complemento resultante. Tener en cuenta si la cifra original se ingresó en positivo o negativo.
8. Por default se dará el resultado en decimal; pero el usuario puede indicar la base numérica en la que desea el resultado con el símbolo '=', al final de la expresión, seguido de la notación de base: B, O, H, D, b, o, h, d, bin, oct, hex o dec. Ej. 6 + 9 = Hex.
9. Se pueden mezclar varias operaciones básicas en una misma expresión. Para esto se debe respetar el orden de precedencia de los operandos * y / vs. + y -. La expresión puede hacer uso de paréntesis bien balanceados (y) para hacer agrupaciones de expresiones anidadas.
10. Variables numéricas: el usuario puede crear variables con nombre alfanumérico. El usuario las puede crear al escribir un nombre seguido del símbolo ':', guardando en dicha variable el resultado numérico de la expresión a la derecha. Ej. var : (4 + 5) * 2.
11. Los nombres de variables son sensibles a mayúsculas y minúsculas, i.e. var != Var
12. Las variables pueden usarse en cualquier expresión, en cuyo caso se usará el valor decimal contenido en la misma.
13. El usuario puede consultar el valor de una variable con solo escribir su nombre y dar enter en el intérprete. Si la variable existe se debe mostrar su valor decimal, si no está definida se debe reportar el error. Si el nombre de la variable está seguido por el símbolo '=' y la indicación de una base según la notación que usen, se deberá dar el valor de dicha variable en la base indicada por el usuario.
14. Cifras de precisión: pueden escoger trabajar internamente con el mínimo de bits necesarios para cada operación o bien redondear a la cantidad superior potencia de 2 mínima necesaria: 4, 8, 16 o 32.
15. Errores, entre otros posibles errores a reportar se deben indicar :

a. Reportar cifras numéricas con símbolos incorrectos. Ej.: 01203B que no corresponde a un número binario.
b. División entre 0
c. Overflow en más de 32 bits en cualquiera de los pasos intermedios de las operaciones o el resultado
d. Errores sintácticos en expresiones mal formadas o paréntesis mal indicados. Ej. 4 + * - 5,) 5- 4, etc.
e. Variables no declaradas

16. Comandos especiales:
    a. #ayuda : brindar un texto explicando las operaciones implementadas por su calculadora y un mini manual que explique el uso general de su calculadora. Si luego del comando #ayuda aparece una palabra clave de la operación u operando, se debe explicar solo esa operación y la forma en que se muestra la salida del procedimiento completo en binario, indicando y explicando sus partes.
    b. #procedimiento : para encender o apagar el modo de operación de la calculadora, en el cual se debe mostrar o no el procedimiento completo de la operación según el detalle siguiente.
    c. #bits : para indicar la cantidad de bits de precisión a usar para operaciones en punto flotante.
    d. #var : mostrará el listado de las variables definidas y sus valores actuales en decimal
    e. #salir : terminar el programa. Se mostrará el listado de las variables definidas y sus valores actuales y terminará la ejecución. Debe mostrar como última salida sus nombres y los datos del curso, periodo, profesor, curso, año, carrera, TEC y periodo.
17. Si el usuario indica que se debe mostrar el procedimiento completo con el comando correspondiente, se deben mostrar las conversiones de todas las cifras de la expresión a binario, se deben mostrar las conversiones realizadas y los valores binarios resultantes. Se deben mostrar paso a paso las conversiones por complemento a base 2 involucradas. Con todos los datos de la expresión en binario mostrados al usuario se debe mostrar la operación en binario correspondiente, paso a paso (sumas, restas, división, multiplicación, punto flotante, complemento). Y brindar el resultado en binario y en la base que lo haya solicitado el usuario.

18. Extras:
    a. Operación de Módulo (residuo de la división)
    b. Operación Potencia **, ej. x**y, x elevado a la y
    c. Las 4 Operaciones básicas con cifras en punto flotante

d. Variables como expresiones: la variable no solo almacena números, sino que puede almacenar un texto como expresión, si la variable es usada, el contenido textual de la misma se sustituye en la expresión actual, este contenido textual podría ser un número, en cuyo caso funcionaría como la descripción original.
e. #var indicando la base de salida deseada. Ej. #var = H, brindará el listado de todas las variables con sus valores actuales en Hexadecimal.

EVALUACIÓN

1. Rúbrica de evaluación:

    - El proyecto se calificará con los siguientes criterios:

        i. 90% - Funcionamiento correcto del programa en NASM según lo solicitado, bajo una estrategia que permita una buena implementación del proyecto.
        ii. 10% - Documentación del trabajo.
        iii. 20% a 30% - puntos posibles distribuidos en varios extras.

2. El proyecto debe resolverse, implementándolo de la mejor manera.

3. Se revisará el código del proyecto, para asegurar que se cumpla con lo solicitado de forma interna y no solo el comportamiento del programa sea como el solicitado.

4. De forma global, se evaluará la presentación del trabajo según los parámetros solicitados, estrategias empleadas y la calidad, la entrega a tiempo del trabajo y la documentación completa correspondiente.

5. Sobre la documentación y presentación:

    a. 2pts - El subject del correo a ser enviado debe ser:
    [Arqui] – Proyecto # 1 – Sus Nombres Completos
    b. 2pts - El correo debe contener de forma separada:
        i. los archivos de texto de los códigos fuentes NASM que permiten la solución y funcionalidad del mismo.
        ii. un archivo PDF con la documentación completa

        No envíe archivos ejecutables o binarios.

    c. La documentación en PDF con el nombre de archivo igual al subject del correo enviado. Esta documentación debe tener un apartado, que indique los pasos a seguir, para poder ejecutar el código (librerías a instalar y otros).
        i. 5pts – La documentación debe incluir una portada con los datos completos: TEC, carrera, sede, curso y código, profesor, periodo, fecha de entrega, número de proyecto,

título del proyecto, nombres completos con número de carnet de la pareja de trabajo y un abstract en inglés en la misma portada.

ii. 5pts – La introducción del documento es una descripción breve del trabajo realizado y herramientas usadas. Como mini-marco teórico incluya las referencias a la documentación del NASM y su página web, al libro Sivarama y las herramientas empleadas, así como otras fuentes de consulta utilizadas.

iii. 10pts – Como desarrollo debe explicar, los procedimientos, rutinas, la lógica que utilizo para resolver el problema. indicar los ejemplos de código que ha usado como guía para el desarrollo de los mismos usando las referencias bibliográficas correspondientes. Explicar el uso y funcionamiento.

iv. 20pts – Análisis de resultados, explicando el trabajo implementado, la forma de realización, funcionamiento, general, ejemplos documentados, problemas presentados, estructuras de datos empleadas, algoritmos usados, etc.

v. 10pts – Una sección de conclusiones y/o observaciones sobre el proyecto.

vi. 10pts – En una sección de Apéndices incluya el código fuente documentado del proyecto y su explicación. Explique la estructura del código empleada en su proyecto, módulos, etc.

d. 10pts – Las líneas del código del NASM deben venir con un comentario que haga referencia a su funcionalidad:

i. i. MOV CL, 100 // inicia el contador para el ciclo de juego

6. La tarea puede realizarse de forma individual o en parejas de 2 máximo.

7. Se debe entregar en digital a más tardar el Sábado 29 de Abril, antes de la media noche. Debe hacerlo de forma simultánea a los correos siguientes y copiarse usted mismo y su compañero de trabajo. Cada día de atraso serán 15pts menos de la nota de la tarea:

a. earias@ic-itcr.ac.cr
b. arirodriguez@ic-itcr.ac.cr

8. Cualquier consulta puede hacerla al foro, o personalmente en clase o al correo del profesor con copia al asistente a los correos anteriores.

9. Durante la revisión del proyecto deben estar presentes ambos miembros de la pareja de trabajo, la no presentación les restará 10pts a cada uno de los ausentes.