|  |  |  |
| --- | --- | --- |
| escudo_uniandes-nuevo_cort.gif | Universidad de los AndesFacultad de Ingeniería Departamento de Sistemas y Computación  Tecnología e Infraestructuras de Cómputo - ISIS 1311  Laboratorio 2 - Ensamblador |  |

**Este trabajo debe ser elaborado individualmente.**

**Puede hacer preguntas al asistente docente y a los monitores para resolver dudas. También puede intercambiar ideas con sus compañeros, pero no puede intercambiar con ellos ningún tipo de código.**

Tenga en cuenta que los laboratorios están diseñados para apoyar su proceso de aprendizaje e identificar y resolver sus dudas sobre los temas estudiados, antes del parcial.

## Objetivos

* Practicar el lenguaje ensamblador.
* Conocer las operaciones de ensamblador para manejo de caracteres y su relación con los operadores de C.

## Actividades

Su tarea es traducir a ensamblador fragmentos de código de programas en C.

**Atención**: solo debe traducir los fragmentos indicados. Lea cuidadosamente la sección de recomendaciones, le ayudará a evitar los errores más comunes.

1. Programa lab2\_p1.c:

a. Compile y corra el programa.

b. Traduzca a instrucciones ensamblador, el cuerpo de la función esDigito. Use los nombres simbólicos para referenciar parámetros y variables locales. Guarde el archivo modificado porque será uno de los entregables.

c. Traduzca a ensamblador el cuerpo de la misma función, esDigito, pero ahora no use nombres simbólicos. Use direccionamiento basado en el ebp. Guarde el archivo modificado porque será uno de los entregables.

2. Programa lab2\_p2.c:

a. Compile y corra el programa.

b. Traduzca a instrucciones ensamblador, en el cuerpo de la función esta. Use los nombres simbólicos para referenciar parámetros y variables locales. Guarde el archivo modificado porque será uno de los entregables.

c. Traduzca a ensamblador el cuerpo de la misma función, esta, pero ahora no use nombres simbólicos. Use direccionamiento basado en el ebp. Guarde el archivo modificado porque será uno de los entregables.

## Recomendaciones

Al momento de hacer la traducción a ensamblador dentro del programa en C, tenga en cuenta las siguientes recomendaciones:

* No modifique las estructuras de datos ni el main de programa.
* Si el código hacer parte de un procedimiento/función en C**, el compilador se encarga de salvar el ebp**. En consecuencia, NO es necesario hacer:

push ebp

mov esp, ebp

Tampoco es necesario recuperar ebp al finalizar la rutina.

**Atención**: de hacerlo dañará el ebp verdadero y no podrá tener acceso a los parámetros y variables locales (leerá posiciones erróneas).

* Dado que las variables locales se declaran en C, el compilador se encarga de separar el espacio en la pila. En consecuencia, **NO es necesario hacer**:

sub esp, <tamaño-locales>

Tampoco es necesario devolver este espacio en la pila al finalizar la rutina.

**Atención**: lo importante con las variables locales es determinar su desplazamiento con respecto al ebp para poder direccionarlas. Para eso, revise las recomendaciones para determinar el desplazamiento en la pila.

* En cuanto a la salvaguarda de registros, puede necesitarla o no; depende de cómo sea su programa.

Dado lo anterior, en el epílogo NO son necesarias las acciones correspondientes: ni recuperar el ebp, ni devolver el espacio de las variables locales. TAMPOCO es necesario escribir el “ret” ese también lo genera el compilador.

Para determinar los desplazamientos en la pila:

* Antes de cualquier cosa, el programa debe estar funcionando bien usando los nombres simbólicos.
* ANTES de hacer CUALQUIER cambio use un printf con las direcciones de TODAS las variables locales y la dirección del primer parámetro. Compile y ejecute.
* Con la información impresa por el printf calcule los desplazamientos de TODAS las variables locales (y los parámetros).
* Solo después de ejecutar todas las recomendaciones anteriores, proceda a reemplazar los nombres simbólicos por sus desplazamientos. A continuación, compile y ejecute. Deje el printf de las direcciones (sirve para determinar las direcciones, pero también para impedir que el compilador elimine las variables locales).

**Atención**: puede ocurrir que en diferentes ejecuciones las direcciones cambien, sin embargo, los desplazamientos se mantienen, así que no hay problema con esto ya que nosotros usamos los desplazamientos y no las direcciones.

* Si ejecuta mal, tome las direcciones impresas en esta ejecución y recalcule los desplazamientos de TODAS las variables locales, reemplace los desplazamientos con los nuevos cálculos (y SOLO eso), compile y ejecute de nuevo.

## D. Entrega

En los 4 archivos que creó incluya la siguiente línea justo antes del encabezado de la función modificada:

// función modificada por: su nombre

A continuación, cree un archivo .zip con los 4 archivos que creó y entréguelo en bloqueneon en el enlace creado para tal fin.