Simulador Von Newman (Lenguaje C)

PROF. KIRSTEIN GÄTJENS S.

"Nadie se ha ahogado jamás en su propio sudor".

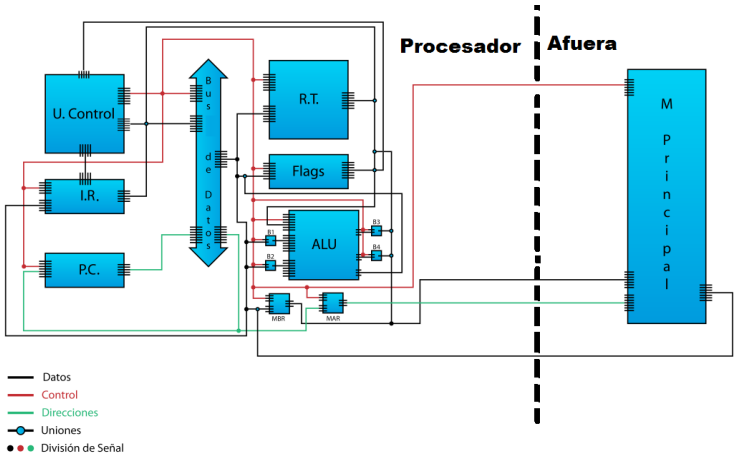
Ann Landers.

Instrucciones generales:

Esta tarea se trata de realizar un simulador que permita probar un microprograma en una arquitectura básica Von Newman.
Debe entregarse antes del viernes 6 de octubre a la medianoche al correo kirstein.eval@gmail.com
Es de carácter individual por lo que cualquier intento de fraude se llevará a las últimas consecuencias.
Se debe desarrollar en el cc para Linux. Pueden usar una biblioteca gráfica como GTK para ayudarse en su trabajo. Si van a utilizar una diferente al GTK deben consultarlo con el profesor.
Entregue un archivo .RAR que contenga el archivo fuente, además de un ejecutable, junto con la documentación solicitada. Si utiliza imágenes o archivos adicionales estos deben estar en una carpeta adicional llamada datos. El nombre de los archivos principales debe ser Simulador-Von-Newman-Apellido-Nombre.XXX donde Nombre y Apellido son los oficiales del curso y XXX la extensión correspondiente (RAR, PDF, C, H, etc.).
El subject del correo es lo usual: TAREA: Simulador Von Newman.
El cuerpo del correo debe contener, como es usual, su nombre completo, número de carné, número de grupo y curso.
Recuerde que la claridad y el orden a la hora de escribir el código y la documentación le ayudarán al asistente a calificarle mejor (y con mayor piedad :).
El valor de la tarea es de 12 resúmenes, numerados del 66 al 77.
Es imprescindible que se trabaje dentro del contexto del paradigma imperativo y NO orientado a Objetos.
Si se utilizan elementos de OO que hagan que su tarea no compile en un compilador estándar de C la tarea no será tomada en cuenta en los rubros de ejecución.
Bono por soledad: Al ser una tarea mucho más interesante, de las que a uno le dan ganas de compartir con un compañero, se les dará un bono por soledad por no poder compartir con nadie del 15% de la nota que obtengan en la factura final. La parte divertida es que aquellos que quieran trabajar en parejas pueden sacrificar el bono por soledad, pero necesitan de reportarse al correo del profesor con el tag: PAREJA: en los siguientes tres días hábiles. Ambos deben reportarse para que sea válida la pareja. Si al final su pareja no trabajó es aceptable cambiarla de nuevo por el bono. (sí, la vida es así de cruel...) pero todo lo que entreguen de tarea debe ser hecho por ustedes mismos y nada por su pareja. En clases discutiremos pormenores al respecto.

Simulador Von Newman

Un Simulador Von Newman es una interfaz que permite probar microprogramas diseñados para la arquitectura Von Newman típica. Utilizaremos la que estudiamos en el curso de Fundamentos. El diagrama de bloques se muestra en la imagen de la derecha, pueden bajar la imagen original del foro de FOC. Va a haber algunas instrucciones de ASM ya definidas en la interfaz, pero la idea es que sea posible definir por parte del usuario más instrucciones de ASM y cargar y probar programas para ese ensamblador.



Lo primero y más importante que deben entender (si es que no lo hicieron en FOC o no lo recuerdan) es que aquí habrá dos niveles de Tanemmbaun involucrados: La microprogramación y la programación en ensamblador (ASM). Las instrucciones de microprogramación básicas

se enlistan y explican más abajo y se espera que su funcionamiento se programe en el lenguaje de alto nivel C. Las instrucciones de ASM se deben programar en el lenguaje de las microinstrucciones y más adelante hablaremos acerca de cuáles de ellas deben entregar funcionando con qué tipos de modos de direccionamiento.

Como es importante referenciar a la arquitectura le llamaremos la AFOC (Arquitectura de Fundamentos de Organización de Computadores).

La interfaz debe permitir ver el diagrama de bloques (mapa de la AFOC) y poder de alguna forma cómoda ir ejecutando paso a paso las microinstrucciones o ejecutar de corrido todo un microprograma de una instrucción ASM. En cada paso debe ser posible ver el contenido de cualquiera de los registros de la arquitectura, así como de la memoria principal.

Recuerde que el principal microprograma que rige el funcionamiento de ejecución de toda instrucción de ASM es el ciclo de Fetch. Deben utilizar el que estudiamos en fundamentos que tenía cinco etapas. Este debe ser el que se ejecute todo el tiempo para cada ejecución de una instrucción de ensamblador. Este debe estar previamente microprogramado y alambrado en la UC, pero debe ser posible ir observándolo ejecutarse microinstrucción a microinstrucción.

Deben de microprogramar (estas ya deben existir en el diseño) al menos las instrucciones de ASM siguientes: mov, add, cmp, jz, jmp, out, in, sti y cli (todas se explican más abajo). Los registros de trabajo que usaremos serán el AX, BX, CX y DX y sus correspondientes partes High y Low de cada uno. Los modos de direccionamiento que deben programar son: inmediato, registro, directo e indexado.

La memoria principal tendrá solo 256 celdas de memoria y se pueden indexar de forma directa o mediante las partes del BX. Recuerden que el BH y el BL son de 8 bits cada uno. En cada celda se puede almacenar o una instrucción o un número de 16 bits. Una instrucción tiene cuatro partes: el código de operación, códigos de dos operandos (fuente y destino) y un posible número adicional de 16 bits al que llamaremos el cuarto dato.

La ALU debe tener predefinido (y programado en C) las siguientes operaciones aritmetico-lógicas que deben funcionar con las banderas solicitadas: Suma, Resta, Multiplicación, División entera con módulo, and, or, xor, not, shl y shr.
El registro de banderas debe tener solo 4 banderas: La Carry Flag, la Sign Flag, la Zero Flag y la Interrupt Flag.

Cada instrucción debe de codificarse adecuadamente. Las instrucciones primitivas deben estar codificadas a partir del código 0 y hasta donde se necesite. Cada instrucción debe saber cuántos operandos tiene (0, 1 o 2) y si requiere de datos adicionales. Ver ejemplo de codificación del mov más adelante.

Funcionalidades del simulador:
+ Acerca de y ayuda clásicos de todo software y para salvar pts si se pone muy espeso algo del proyecto.
+ Cargar y Salvar a disco una AFOC. Debe permitir almacenar en memoria secundaria y recuperar todos los detalles definidos por el usuario del simulador (instrucciones de ASM). El formato de los archivos para ser almacenado puede ser a su conveniencia incluyendo varios archivos de texto.
+ Resetear una AFOC, restaurar a los valores inciales a la arquitectura básica y limpiar toda memoria y registros.
+ Definir una nueva instrucción de ASM. Debe ser posible ponerle nombre y n-mónico, codificarla y crear sus datos y escribirle un microprograma asociado a cada combinación de modos de direccionamiento. La interfaz para escribir el microprograma queda a su criterio, pero debe ser cómoda. Analicen la comodidad de cargar un archivo de texto vs el trabajo de parsearlo o usar botones y ventanas para recibir los datos de entrada.
+ Load ASM: Debe ser posible cargar un programa de ASM a partir de una posición de memoria. Lo esperable es que se pueda escribir en un archivo de texto y se cargue a memoria. Sin embargo pueden tener en lugar de eso la funcionalidad siguiente.
+ Edit ASM: Se debe poder escribir y salvar en disco un programa de ASM. Interfaz de escritura a su criterio, pero debe permitir utilizar tanto las instrucciones de ASM primitivas como las definidas en la AFOC actual. Esta funcionalidad solo se debe tener si no se escoge procesar un archivo de texto.
+ Ejecutar la AFOC. Debe ser posible resetear el PC o asignarlo manualmente para comenazar a ejecutar. Recuerde que se debe poder ejecutar cada instrucción de ASM paso a paso. Dentro de cada microprograma se puede ejecutar de corrido o microinstrucción por microinstrucción.
+ Consultar/modificar el valor de cualquier registro en cualquier paso de ejecución.

+ Interrumpir al procesador. Debe ser posible tener un botón en la interfaz que indique que se creo una interrupción. Esta debe encolarse y ser atendida en el momento oportuno. Para atender una interrupción basta con sacar una ventana que indique que ese número de interrupción está siendo atendida.
+ Salir decentemente del simulador.

Instrucciones de ASM mínimas:
Mov: Clásico mov que permite mover datos entre lugares de la AFOC. Trabaja con todos los modos de direccionamiento mencionados. Ejemplos de uso:
mov ax, bx
mov al, -5
mov [bl], ch
mov ah, [75]
Out e **In** son dos intrucciones que programaremos para que el programador en ASM se pueda comunicar con el usuario del simulador. Out se espera que trabaje enviando a una ventana de salida el contenido de un registro y que In pida en una ventana un número y que se almacene en un registro. Ejmeplos de uso:
Out Ax
In Bh
Jmp es el clásico salto incondicional. Para simplificarnos no se trabajará con etiquetas sino que será el salto a una posición directa de la memoria. Ejemplo:
jmp 17
Jz: Es el salto condicional, donde se bifurca si la zf está encendida y no si no lo está. Ejemplo:
jz 15
cmp: Es la instrucción que compara dos elementos. Recuerden que se implementa restando ambos elementos y modificando las banderas y olvidándose del resultado. Permite los mismos operandos que el add.
cli y **sti** apaga y enciende la Interrupt flag. Respectivamente.

Microinstrucciones mínimas de la AFOC:
X<-Y es la que permite mover datos del registro X al registro Y.
ALU: op ejecuta la operación correspondiente de la ALU (add, sub, mul, div, and, or, xor, not, shl o shr).
MEM: op ejecuta la operación de acceso a la memoria correspondiente (read o write)
TEST: flag, N bifurca a la instrucción N del microprograma si la bandera flag está encendida.
In abre una ventana y solicita un número que dejará almacenado en el MBR.
Out En la ventana de salida despliega el contenido del MBR.

Deben diseñar claramente las codificaciones de las instrucciones primitivas y a partir de allí permitir que el usuario del simulador agregue más instrucciones de ASM. Puede ser que una instrucción requiera de más de un formato en su codificación, en este caso deben tener códigos de instrucción diferentes y microprogramas de ejecución distintos.

Ejemplo de codificación: **MOV**

Formatos válidos del mov son dos.

1. Movimiento de un Reg/Mem hacía un Reg/Mem
Código de instrucción 0. Tiene dos operandos. Ambos se codifican en los campos correspondientes con la tabla de codificación de R/M. El cuarto dato se usa como dirección en los casos de acceso directo.
Recuerde que no se debe permitir dos accesos a memoria a la vez porque solo tendremos un subciclo indirecto en el ciclo de fetch, además de que solo tenemos un único cuarto dato.

2. Movimiento de un inmediato hacía un Registro.
Código de instrucción 1. Tiene solo un operando destino. Codifica el inmediato en el número adicional (cuarto dato). Como usa ese cuarto dato para el dato inmediato no se puede permitir usarlo para acceso a memoria, por lo que no hay un formato que permita mover inmediatos a memoria directamente.

Tabla R/M:

AX = 0	BX = 1	CX = 2	DX = 3
AL = 4	BL = 5	CL = 6	DL = 7
AH = 8	BH = 9	CH = 10	DH = 11
[dir] = 12	[BL] = 13	[BH] = 14	

Piensen que esto se puede complicar tanto como uno quiera, pero lo estamos tratando de mantener en un nivel que sea aceptable. Eliminamos el uso de procedimientos y de una pila para trabajar con ellos. Aún así tenemos la esperanza de que esta sea una versión aceptable para poder probar microprogramas y programas en este ensamblador.

Documentación

Por ser un proyecto no tan pequeño, la documentación debe ser aceptable. Debe ser entregada en un PDF aparte. Debe incluir los siguientes elementos usuales para tareas de este tamaño:

+ Portada de página completa.

+ Análisis de Resultados siguiendo el formato usual de mis cursos.

Se debe dividir en partes todo lo que hay que hacer y asignarle una calificación con letra que va de la A a la E.

Donde la A representa: Concluido con éxito.

Donde la B representa: Concluido con problemas pequeños. Requiere aclaraciones.

Donde la C representa: Concluido con problemas mayores. Requiere aclaraciones.

Donde la D representa: Concluido solo en diseño. Codificación quedó incompleta. Requiere aclaraciones.

Donde la E representa: Tarea no implementada.

+ Un pequeño manual de usuario que explique claramente como se debe instalar y usar el software entregado. Debe ser muy explícito en cuanto a la forma de usar todas las funcionalidades del simulador.

+ Deben documentar claramente toda la codificación de todas las instrucciones de ASM primitivas y todos los microprogramas que están previamente alambrados en el simulador.

+ No olviden que el código interno en cada función debe estar bien documentado. Tomando en cuenta precondiciones, poscondiciones, parámetros, etc.

Suerte!!