Universidad Eafit

Departamento de Informática y Sistemas

ST0244 – Lenguajes de programación

Proyecto 2: Simulador de ejecución de programa en procesador Hack

Aplica para aquellos que cumplieron con las 6 historias del proyecto 1

Si usted es de los que no terminó el proyecto 1, vaya a la página 6

Adaptación de los proyectos de Máquina Virtual de la plataforma educativa Nand2Tetris para el curso de Lenguajes de programación.

Introducción El Assembler es un lenguaje de programación que, aunque a primera vista parezca ajeno y extraño a la programación normal, explica a detalle el comportamiento de la máquina. Y visualizar su comportamiento, como fluye la información en el procesador, cómo se mueven los registros y qué se afecta, es una forma adicional de comprender cómo funciona ese mundo de la infraestructura de hardware asociada a un programa.

Usando nuevamente la plataforma Hack, vamos a construir un simulador del procesador Hack. Partiendo del archivo que deja el ensamblador de la práctica 1, vamos a construir una herramienta en modo texto que nos permita simular la ejecución del programa Hack paso a paso.

Objetivo Escribir en C++ un programa que lea el archivo ensamblador de Hack, dibuje en pantalla una simulación del procesador y lo ejecute paso a paso hasta terminar. Se debe visualizar el contenido de los registros A y D, parte de la memoria (donde la estemos ejecutando), obtener entradas desde el teclado y ver el contenido de la memoria RAM.

Análisis nivel 0 En la figura 1 se describe de manera general el proceso a implementar:

Hay tres formas de describir el comportamiento esperado de su programa:

1. Cuando se ejecuta el Hack Runtime, debe leer un archivo de disco con un nombre (por ejemplo, Prog.hack) que contiene un programa válido en código binario de Hack correcto. La forma de invocar el programa con un archivo llamado programa.hack es:
   HackRT programa.hack
2. El programa en binario se debe cargar así cómo está, en un elemento que simule la memoria del procesador Hack. Recuerden que la memoria de programas del Hack es de 32768 posiciones de memoria, la memoria de datos es de 24576 posiciones, ambas de 16 bits.
3. Los registros PC, A y D se resetean a cero y se arranca la ejecución paso a paso del programa hasta que termina (se intenta ejecutar una posición de memoria que no tiene

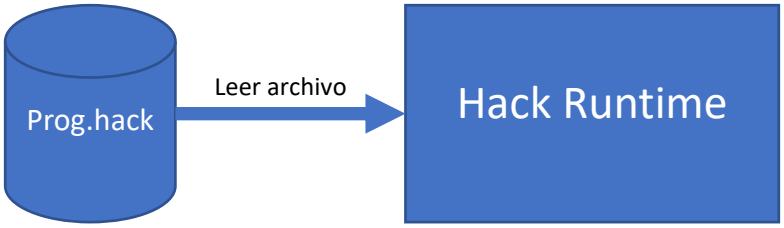

Figura 1. Proceso general del programa

datos), reflejando en los registros A y D y en la posición de memoria indicada por el registro A los nuevos valores luego de ejecutar cada instrucción.

Análisis nivel 1 El ensamblador debe ejecutarse usando un comando en la línea de comandos, como por ejemplo "HackRT fileName.hack", donde *fileName.hack* es el archivo de entrada del HackRT, es decir, el nombre de un archivo de texto que contiene comandos de código assembler de Hack. A partir de ese archivo, se llena la memoria de programa desde la posición 0 hasta donde termine, en cualquier caso, nunca va a ser mayor a 32768 líneas porque cada línea en el archivo con extensión .hack es una instrucción que va en una posición de memoria.

Luego, aparece en pantalla el contenido de los registros A, D, PC, la memoria de programa y la memoria de datos.

Cuando se empieza a ejecutar el programa, el contador de programa (PC) arranca en 0 y lee la instrucción de la primera posición de la memoria de programa, determina si es una instrucción tipo A o C y ejecuta la acción requerida. Una vez se ejecute la acción, el PC se incrementa en 1, excepto en los casos en que la instrucción a ejecutar sea un salto, donde el nuevo valor del registro PC es reemplazado por el contenido del registro A.

Implementación propuesta

a. **Clases a implementar**
Se propone el siguiente diagrama de clases (ver figura 2).

Byte Almacena en un campo llamado strContenido un dato de 16 caracteres con 0's y 1's en ASCII. Tiene los siguientes métodos:

1. leer: retorna el string almacenado en strContenido.
2. escribir: toma el string que venga como parámetro y lo guarda en strContenido.

Memoria Crea un arraylist de objetos Byte con la cantidad de posiciones requeridas. Tiene los siguientes métodos:

1. leer: retorna un string con el contenido de una posición de memoria enviada como parámetro. Si la posición de memoria solicitada no existe, se lanza una excepción para indicar error al CPUHack y que se aborte la ejecución del programa.

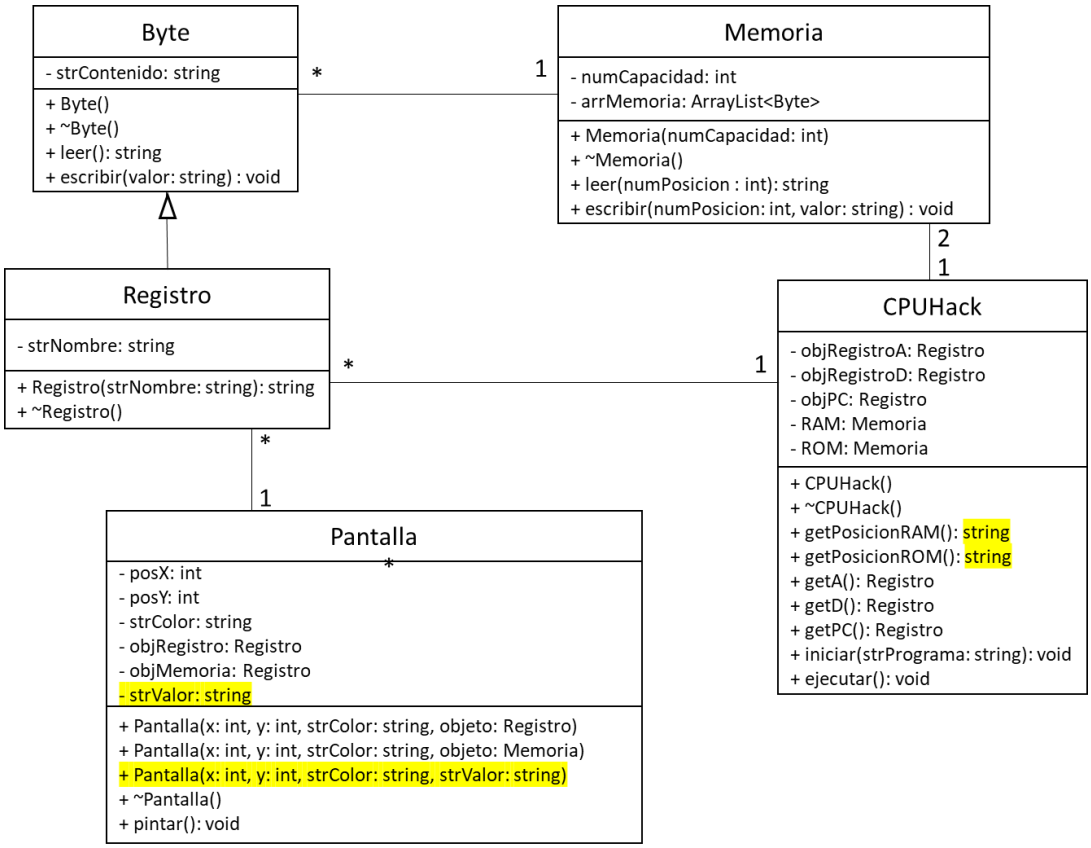

Figura 2. Diagrama de clases propuesto.

Nota: Las marcadas en amarillo, cambiaron o son nuevos métodos o atributos a implementar.

2. escribir: modifica una posición de memoria con un string que se envía como parámetro al método. Si se trata de acceder una posición de memoria que no existe, se lanza una excepción para indicar error al CPUHack y que se aborte la ejecución del programa.

Registro Hereda de la clase Byte agregando un campo llamado strNombre para agregar el nombre del registro.

Pantalla Relaciona un objeto tipo Registro o un valor almacenado en una posición de memoria con una posición en pantalla donde se va a pintar. Tiene dos atributos (posX y posY) para indicar la posición donde se va a pintar el objeto y un string que indica el color que se debe colocar en ese registro.

Tiene los siguientes métodos:

1. pintar: imprime en pantalla en la posición indicada el contenido del objeto con los colores indicados.

CPUHack Contiene la descripción de registros y memoria que la CPU tiene asignada para realizar la ejecución de un programa en Hack. Tiene los siguientes métodos:

1. getPosicionRAM: devuelve un string con el contenido de la posición de la memoria RAM indicada por el registro A.

2. getPosicionROM: devuelve un string con el contenido de la posición de memoria ROM indicada por el registro PC.
3. getA: devuelve un objeto tipo Registro con el contenido del registro A.
4. getD: devuelve un objeto tipo Registro con el contenido del registro D.
5. getPC: devuelve un objeto tipo Registro con el contenido del registro PC.
6. iniciar: lee el programa que se envía como parámetro desde disco, crea los objetos RAM, ROM, los registros y deja todo listo para que el programa empiece a ejecutarse.
7. ejecutar: Ejecuta instrucción por instrucción modificando los registros de acuerdo a la instrucción que esté ejecutando en este momento. Cuando termina el programa, notifica en pantalla que el programa finalizó. Por otra parte, si se presenta un error de ejecución por alguna excepción generada, muestra en pantalla un mensaje.

Mecánica de la implementación Los estudiantes implementan todas las clases excepto la clase *CPUHack*. Esta será implementada por el docente.

Fecha de entrega La práctica se debe entregar el sábado 27 de noviembre antes del medio día.

Sustentación Previamente, el docente ha verificado la originalidad del código de cada equipo. Posteriormente, en presencia de los estudiantes se usan los archivos entregados por cada equipo y compila la solución completa con la clase del profesor usando el makefile. Luego ejecuta los programas de prueba y compara los resultados de la ejecución con los resultados esperados y obtiene una parte de la nota. En caso de que el programa no compile, se darán hasta 3 oportunidades de realizar modificaciones para solucionar el inconveniente presentado. Pasadas las tres oportunidades, se valorará por el docente el error y la corrección pertinente de acuerdo con el siguiente criterio de calificación.

Criterios de evaluación El cumplimiento de cada historia tiene los porcentajes que se pueden ver en la tabla 1 ubicada en la página 5.

Integrantes para la práctica Se puede hacer en equipos hasta de tres personas máximo. Se debe notificar al profesor máximo hasta el 19 de noviembre a las 10 de la noche.

Archivos adicionales Para efectos de pruebas, el sábado 20 después del medio día se dejará en Teams los archivos necesarios para que realicen las pruebas de cada clase a implementar. Se dejarán pruebas individuales de cada clase y un “simulador” de la clase CPUHack para efectos de pruebas.

"It is better to fail in originality than to succeed in imitation." -- Herman Melville.

¡Muchos éxitos!

Criterios	Porcentaje
Se implementa correctamente* la clase Byte	10%
Se implementa correctamente* la clase Memoria	20%
Se implementa correctamente* la clase Registro	20%
Se implementa correctamente* la clase Pantalla	20%
Sustentación (integración con la clase CPUHack del profesor). Se otorga una de las siguientes opciones:	30%
Funciona correctamente sin errores o se corrigen los errores presentados en los tres intentos de corrección	30
No se logra corregir en los tres intentos y el profesor considera que estaba en camino a ser implementado (había como mínimo, implementación en todos los métodos)	20
No se logra corregir en los tres intentos y faltaba mucho código por implementar	10
No se logra corregir, se elije al azar a un integrante del equipo (si es más de una persona el grupo) y se le pide hacer una modificación al código para agregar una funcionalidad o subsanar un error que el profesor detectó y es fácil de corregir y funciona correctamente.	Hasta 30 puntos, a criterio del profesor en el momento de la sustentación.

Tabla 1. Porcentajes de los criterios de evaluación

* La palabra *correctamente* en este contexto significa que se implementó la clase con la descripción descrita en el diagrama de clases (atributos y métodos con el mismo nombre del diagrama). En caso de requerirse un cambio en la clase por un error de diseño detectado por el profesor durante la implementación, este informará a todos los estudiantes para que realicen la modificación respectiva.

Universidad Eafit

Departamento de Informática y Sistemas

ST0244 – Lenguajes de programación

Proyecto 2: Reimplementando el proyecto 1

Introducción Partiendo de los archivos que tenía en el canal de Teams del proyecto 1, debe realizar lo siguiente:

Historia 1 Copiar los archivos entregados al nuevo canal creado para el proyecto 2.

Historia 2 Realizar las modificaciones necesarias a cada historia para que funcione correctamente. Las historias que no deben implementar en este proyecto 2 son la 4 y la 7 del proyecto 1.

Fecha de entrega La práctica se debe entregar el sábado 27 de noviembre antes del medio día.

Sustentación Previamente, el docente ha verificado la originalidad del código de cada equipo. En su caso, se evaluará la práctica de acuerdo con el siguiente criterio de calificación:

Historia	Porcentaje
1 Proyecto 2	5%
1 Proyecto 1	5%
2 Proyecto 1	5%
3 Proyecto 1	5%
5 Proyecto 1	15%
6 Proyecto 1	15%
Pruebas con archivos adicionales	20%
Sustentación	30%

Tabla 1. Porcentajes de los criterios de evaluación

Su sustentación consistirá en una adición o modificación elegida al azar por el profesor para determinar su nivel de entendimiento del código presentado y de lo visto durante el semestre en C++. Entre las adiciones o modificaciones (a modo de ejemplo) pueden estar:

- Agregue un método a la clase Parser para validar una instrucción nueva de Hack
- Modifique el Parser para que solo identifique letras. Debe marcar error si encuentra un número.
- Modifique el código de la lectura del archivo para que cada línea que lea salga en un archivo nuevo de salida llamado cambiado.txt donde cada línea leída se le agregue un número que indica cual línea es, se deje un espacio y se agregue el código leído. Por ejemplo

  @20 quedaría 1 @20

  D=A 2 D=A

Y así otras más similares.

"It is better to fail in originality than to succeed in imitation." -- Herman Melville.

¡Muchos éxitos!