# CC4301 Arquitectura de Computadores. Semestre Primavera 2016. Tarea 2

Profesor: Pablo Guerrero

Entrega: Martes, 22 de noviembre de 2016, 23:59 hrs.

En esta tarea desarrollaremos una versión extendida del compilador de la minitarea 6. Este compilador opera para una versión simplificada de la máquina abstracta SECD[1] utilizada en el curso de Lenguajes de Programación. Dicha máquina utiliza el stack para encolar instrucciones y el resultado parcial de su ejecución. **Su programa debe recibir una lista de instrucciones SECD y generar un código en Assembler ARM** que ejecute las instrucciones secuencialmente (en una máquina ARM). En esta versión de la máquina abstracta, existe la posibilidad de definir funciones y de luego aplicarlas. Las funciones siempre retornan un valor entero. Las instrucciones SECD disponibles en la máquina que implementaremos son:

- `(INT_CONST n)`: Apila el entero `n`.
- `(ADD)`: Toma los dos elementos en el tope de la pila y los suma, reemplazando en la pila dichos elementos por el resultado de la suma.
- `(SUB)`: Toma los dos elementos en el tope de la pila y los resta, reemplazando en la pila dichos elementos por el resultado de la resta.
- `(FUN inst_list)`: Definición de una función. Recibe una lista de instrucciones, inst_list, que corresponde a la implementación de la función, como argumento. La función debe quedar implementada antes del `main` y debe apilarse un puntero a dicha función. Lo primero que debe hacer la función al ejecutarse es apilar su argumento para que lo usen sus instrucciones.
- `(APPLY)` Ejecuta una función. Usa el elemento en el tope del stack como argumento para el llamado a la función, y el elemento siguiente en el stack como puntero a la etiqueta para ejecutar la función. Recibe el valor retornado por la función y lo apila.
- `(RETURN)` Retorna la función. Para ello, desapila el valor a retornar y lo retorna utilizando el mecanismo de retorno de funciones de assembler.

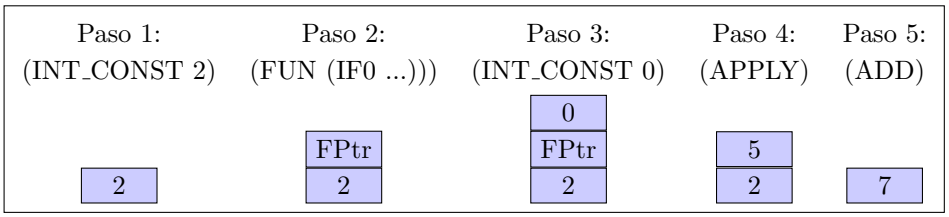
[1] Más información en: La página de SECD en Wikipedia

- `(IF0 tb fb)` Desapila el primer elemento del stack. Si es igual a cero, ejecuta la lista de instrucciones `tb`. En caso contrario, ejecuta la lista de instrucciones `fb`.

Un ejemplo de una lista de instrucciones es el siguiente:

```
((INT_CONST 2)
(FUN (IF0 ((INT_CONST 5)) ((INT_CONST 7))) (RETURN))
(INT_CONST 0) (APPLY) (ADD))
```

La ejecución de dicha lista debería producir en el stack los siguientes estados, paso a paso:




En el paso 2, del esquema anterior, se apila `FPtr`, que es un puntero a la función. Dicho puntero, corresponde a la etiqueta de la implementación de la función. En el paso 4, se ejecuta la función con parámetro `0`, por lo que esta devuelve `5`. Dicho resultado se apila.

Para apilar y desapilar los punteros a funciones, puede obtener un puntero a una etiqueta y guardarlo en un registro. Luego puede saltar a dicha etiqueta usando el valor de un registro. Por ejemplo,

```
label:
    [...]
    LDR R7, =label   %R7 = puntero a label
    [...]
    BL  R5   %llamada a la subrutina apuntada por R5
```

Para permitir el uso de funciones anidadas, se recomienda hacer una primera pasada de compilación donde se compilen todas las funciones y se guarden sus etiquetas en una tabla y una segunda pasada (análoga al proceso de linking de un compilador) donde se reemplacen los nombres de las respectivas etiquetas en los lugares donde se utilizan.

Para generar el stack necesario para la ejecución de la lista de instrucciones SECD, el programa compilado debe utilizar el manejo de stack provisto por ARM y respetar la convención del llamador con stack descendiente.

Al finalizar de ejecutar el código de las instrucciones compiladas, debería quedar en la pila un único entero con el resultado de la ejecución. El compilador debe agregar al final del código Assembler un trozo que desapile este resultado y lo entregue, ya sea en un archivo o en alguna salida visible.

**P1.-** Implemente el compilador previamente descrito. Para ello, puede usar Scheme, C ó C++. Su programa debe leer la lista desde un archivo, recorrerla, y por cada instrucción SECD, agregar un trozo de código Assembler ARM en el programa de salida, el cual debe ser guardado en un archivo. **En el caso de las funciones, sus definiciones deben agregarse antes del programa principal, con sus respectivas etiquetas**. En el programa principal, la definición de la función debe provocar el apilamiento del respectivo puntero a la etiqueta de la función. Por lo tanto, probablemente, deba recorrer más de una vez la lista de instrucciones (considerando que dentro de las definiciones de funciones pueden haber otras funciones anidadas, las cuales deben ser llamadas de la misma manera).

Si utiliza Scheme, puede utilizar la función `(display-to-file)` para guardar en un archivo. El siguiente es un ejemplo de su uso:

```
(display-to-file "Hola Mundo!"
                 "file.txt"
                 #:mode 'text
                 #:exists 'replace)
```

Si utiliza C ó C++, use la librería `sexpr`[2] para leer la lista de instrucciones a una estructura fácil de recorrer.

Utilice la misma máquina ARM que en la minitarea 5 (sea ésta real o emulada) para probar el código obtenido desde su compilador.

**P2.-** Elabore algunas secuencias de prueba de instrucciones SECD de manera de testear el funcionamiento de su compilador. Ejecute dichas secuencias y chequee que el resultado sea el esperado. Dentro de estas secuencias, al menos debe haber un uso de cada instrucción y una definición y aplicación de función anidada.

Entregue en u-cursos:

- Un documento en pdf que contenga:
  - Una explicación de la solución implementada.
  - Una descripción de las pruebas diseñadas, lo que pretende probar cada una de ellas, y un análisis de los resultados obtenidos.
- Los archivos de código fuente, en el lenguaje utilizado, con comentarios para poder entender clara y detalladamente su funcionamiento.

**Descuentos:** Una décima por hora, un punto por día de atraso.

[2]Descargar en sourceforge. Más información en: La página web del proyecto `sexpr`