Instituto Tecnológico de Costa Rica
Escuela de Computación
Primer Semestre de 2020

IC1301 Arquitectura de Computadoras
Proyecto #1

Descripción general: Ensamblador sencillo de dos pasos de algunas instrucciones ensamblador del procesador TEC-IC.

Objetivos didácticos: Los estudiantes

1. Desarrollarán destrezas básicas en el uso de las expresiones regulares para la descripción de lenguajes formales L_3, o de elementos de lenguajes L_2.
2. Entenderán mejor la relación entre lenguaje ensamblador y lenguaje máquina
3. Adquirirán una mejor comprensión del proceso de traducción de lenguaje ensamblador a lenguaje máquina
4. Adquirirán una comprensión básica del proceso de traducción de un compilador de un lenguaje de alto nivel a lenguaje máquina.

Temas del curso cubiertos:

1. Características formales de los lenguajes ensambladores.
2. Expresiones regulares, autómatas determinísticos finitos.
3. Generación de lenguaje máquina
4. Formato de las instrucciones máquina.

Descripción específica:

1. *Proceso que debe hacer el programa:* Específicamente, el programa a desarrollar recibirá un programa fuente ASCII de código ensamblador TEC-IC, y generará un archivo de salida binario con el código máquina .

2. *Entradas que debe recibir:* el archivo con el código fuente.

3. *Interfaz con el usuario:* La interfaz consistirá de una ventana simple que permita al usuario escoger el archivo fuente de una lista en el directorio actual, con la posibilidad de cambiar de directorio o cancelar, un botón de “traducir”, algún medio de salir sin traducir y algún medio de mostrar algún error encontrado en la entrada.

4. *Formato de la entrada:*

   `.program` <nombre del programa>
   `.const` <lista de constantes de usuario con sus valores>
   `.text` <lista de instrucciones>

   La lista de instrucciones tendrá el formato [<etiqueta>:] [↓|ε] <instrucción>
   Se permitirá el uso de comentarios, un comentario tiene la sintaxis “;<hilera>”, donde tanto el “;” como la hilera serán ignorados. Un comentario podría ser solo parte de una linea, o toda la linea.
   Para permitir al usuario el dar formato a su programa se podrá sustituir cualquier ocurrencia de un espacio en blanco por una secuencia no nula de (□|→)
   También por propósitos de formato se permitirá que halla lineas nulas.

5. *Formato de las salidas:* en caso de que el programa tenga una sintaxis correcta, la única salida sería un archivo binario “ejecutable”. El formato binario de las instrucciones máquina se brinda más adelante en este documento.

6. *Restricciones del proyecto*
   1. El lenguaje de desarrollo deberá ser Python, para que los estudiantes aprovechen el conocimiento del lenguaje que adquirieron en cursos anteriores, y, para aquellos estudiantes que consientan en ello, que su programa sea usado por otros estudiantes para mejorar el ensamblador y adquirir destrezas más realistas sobre actualización de código.
   2. Deberá reconocer como correcto el programa nulo.
   3. En caso de que haya un error de sintaxis, el programa deberá informar al usuario el número de linea y la naturaleza del error.

7. *Manejo de las excepciones encontradas en la entrada*: La estrategia de manejo de errores será la que se conoce en el ambiente de diseño de compiladores como de “pánico”, es decir, en cuanto se encuentre un error de gramática el proceso de traducción se detendrá y se indicará al usuario que se encontró un error, en cuál linea del archivo fuente se encuentra, y cuál es el error encontrado.

8. Ejemplo de entrada:

```
.program Primer ejemplo
;-----------------------------------------
; Programa para ilustrar una entrada posible
; -----------------------------------------

.const  a = 28
        b = 0x2C3
.text
     ciclo_infinito:
          ld R1,a(R0)  ; Carga el puntero inicial
          ld R2,b(R1)
          ld R3,32(R2)  ; Carga el valor verdadero
          add R4,R3,R2
          and R4,R4,R2
     jmp ciclo_infinito
     st R4,a(R0)
```

Salida correspondiente: con un programa de dump se vería la siguiente salida, en binario

```
00000000000000011100 0001 0000 00010
00000000001010110011 0010 0001 00010
00000000000000100000 0011 0010 00010
0100 000 0 0000000 0100 0011 0010 00001
1001 000 0 0000000 0100 0100 0010 00001
11111111111111111111101000 01001
00000000000000011100 0100 0000 00100
```

(se han puesto separaciones artificiales solo para que los estudiantes vean los campos de la instrucción).

Por consiguiente, el archivo de salida contendría

```
00000000000: 00038202 00566422 00040642 40008642  90008841 FFFFFD09 00038804
```

Detalles del lenguaje ensamblador

Instrucciones a implementar:

Instrucción	**IC**	**Sintaxis**
ld	2	`ld  Rd,Dd(Rb)`
st	3	`st  Rd,Dd(Rb)`
add	1 (f = 4, op = 0)	`add  Rd, Rf1, Rf2`
adc	1 (f = 4, op = 1)	`adc  Rd, Rf1, K`
and	1 (f = 9)	`and  Rd, Rf1, Rf2`
jmp	9	`jmp  label`
testl	1 (f = 5, cond = 5)	`testl Rf1,Rf2`

Formato de las instrucciones máquina:

Tipo A:

Bits	Campo	Función
0..4	IC [01]	Código de instrucción
5..8	Rf2	Registro fuente 2
9..12	Rf1	Registro fuente 1
13..16	Rd	Registro destino
17..22	K	Constante entera
23	Tipo	Tipo del segundo operando 0: registro fuente 2 1: constante
24..27	Condición	Condición a verificar (solo para inst. de `test`)
28..31	f	Función (extensión del código de instrucción)

Tipo I

Bits	**Campo**	**Función**
0..4	IC	Código de instrucción
5..8	Rb	Registro base
9..12	Rd	Registro de dato
13..31	D_d	Desplazamiento a partir de la dirección base (Segmento de datos).

Tipo J

Bits	**Campo**	**Función**
0..4	IC	Código de instrucción
5..8	D_c	Desplazamiento a partir del valor actual del PC (Segmento de código)

Documentación

1. Descripción del algoritmo del programa.
2. Descripción completa del lenguaje implementado, mediante expresiones regulares.
3. Gráficos de los autómatas implementados

Sesiones de guía:

1.
    1. Fecha: 10/Marzo
    2. Trabajo esperado: Gramática del lenguaje (expresiones regulares), diseño general del ensamblador (3 niveles)
2.
    1. Fecha: 17/Marzo
    2. Trabajo esperado: Diseño de la interfaz, diseño de al menos un reconocedor gramatical
3. Fecha: 24/Marzo. Entrega

Desglose de calificación:

Documentación [20%]

1. Descripción del algoritmo: 7%
2. Descripción del lenguaje: 7%
3. Gráficos de autómatas: 6%

Interfaz de usuario [10%]

1. Contiene todos los elementos solicitados: 3%
2. Todos los elementos funcionan correctamente: 4%
3. Es fácil de usar: 3%

Ejecución [70%]

Manejo de casos típicos: 15% (5% cu)

- Caso 1
- Caso 2
- Caso 3

Manejo de situaciones anómalas: 30% (5% cu)

- Código de instrucción inexistente
- Error de sintaxis 1: faltan operandos
- Error de sintaxis 2: operando incorrecto o irreconocible
- Registro inexistente
- Constante fuera de rango
- No maneja negativos (ld, st, adc)

Manejo de casos extremos: 15%

- reconoce el programa nulo: 5%
- reconoce lo máximos de las constantes: 10%

Publica opcionalmente la tabla de símbolos: 10%