ATR: Traductor de código en formato ensamblador a código máquina

Creado por:
Alexis Rodríguez Casañas
Alberto Martín Núñez
04.04.2016

Introducción

Con el objetivo de facilitar la escritura y lectura de los programas creados para la CPU se ha implementado una herramienta de traducción que intenta acercarse a la programación en MIPS. Se trata de un programa/script escrito en Ruby que evaluará y traducirá el código siguiendo unas mínimas reglas.

Repertorio de instrucciones:

A diferencia de MIPS, las palabras de las instrucciones no van separadas por comas sino únicamente por un espacio en blanco. A continuación se presentan las instrucciones disponibles. Los registros van desde $0 hasta $15 y los puertos desde P0 hasta P3.
Las instrucciones siguen el formato “DESTINO, OPERANDO B, OPERANDO A”. En la mayoría de las operaciones esto es irrelevante, pero es necesario tenerlo en cuenta en el caso de la resta. En cuanto a las instrucciones que hacen referencia a un puerto, nótese que éste siempre va al final.

Las instrucciones disponibles con su correspondiente formato son las siguientes:

LI 5 $1	Carga un 5 en REG1
ADD $1 $2 $3	$1 = $3 + $2
SUB $1 $2 $3	$1 = $3 - $2
AND $1 $2 $3	AND lógico
OR $1 $2 $3	OR lógico
ONC $1 $3	$1 = Ca1 de $3
TWC $1 $3	$1= Ca2 de $3
ASG $1 $3	$1= $3
JL etiqueta	Salto incondicional
BZ etiqueta	Salto si la última operacion fue = 0
BNZ etiqueta	Salto si la última operación fue != 0
RELJ -5 / 5	Salto relativo. Retrocede / avanza 5 instrucciones
GO etiqueta	Salto a subrutina
RETURN	Retorno de subrutina (nótese que debe escribirlo el programador)
WIP 3 P0	Escribe un 3 en el puerto de salida 0
WDP $1 P0	Escribe el contenido de REG1 en el puerto de salida 0
RDP $1 P0	Lee el puerto de entrada 0 y lo escribe en REG1

Comentarios:

Es posible escribir comentarios en el código precedidos del carácter "*". Deberán estar en una línea nueva, es decir, no pueden haber comentarios al lado de las instrucciones. Cumpliendo esta regla, puede jugarse como se desee con los comentarios.

```
*creado por Ususario
*fecha
*******inicio******
LI $3 8
ADD $1 $2 $1
ADD $3 $2 $1
*salta a la etiqueta suma
JL suma
*****fin del programa
```

Etiquetas:

Las etiquetas pueden ser cualquier cadena de texto. Pueden incluir números siempre que no estén formadas únicamente por los mismos. A diferencia de MIPS, las etiquetas no van precedidas de dos puntos. Se recomienda escribirlas en minúscula para que contrasten con las instrucciones.

```
1  carga
2  LI $1 3
3  LI $2 2
4  JL suma
5  LI $3 1
6  suma
7  ADD $1 $2 $1
8  BZ carga
```

El código **no puede contener líneas en blanco**. Una posible estrategia si se desean separar instrucciones para mejorar la lectura es jugar con los comentarios, enmarcando o separando instrucciones.

```
LI $3 8
JL suma
*
*
suma
ADD $1 $2 $1
JL resta
************
resta
SUB $1 $1 $3
JL suma
***********
```

Utilizando el traductor:

Con los ficheros *atr.rb* y *dictionary.dic* en el mismo directorio, basta con ejecutar:

ruby atr.rb programa

Esto arrojará un fichero *progfile.dat* en código máquina. Al tener el mismo nombre, si se colocan los ficheros en el mismo directorio de trabajo de Verilog, podremos cargar instantáneamente el código en la memoria de la cpu cada vez que ejecutemos el traductor.
Si el traductor no encuentra una instrucción la traducirá por *xxxxxx* y continuará su ejecución. En ese caso se puede revisar el fichero diccionario.

```
xalex1200:~/workspace $ ruby atr.rb prog.s
Ignorando comentarios...
Sustituyendo etiquetas...
AVISO. No hay traduccion para BQ (traduciendo por "xxxxxx")
xalex1200:~/workspace $
```

Sin embargo, si el traductor no encuentra una etiqueta o el formato de la misma es erróneo no informará de error alguno y el fichero *progfile.dat* no producirá el comportamiento deseado a pesar de tener una apariencia correcta. Dado que no forma parte de los requisitos de la asignatura, el tiempo es limitado y el lenguaje es muy sencillo, se ha considerado que no es viable el esfuerzo/tiempo que es necesario invertir para implementar mecanismos de detección y depuración de errores. Por ello, el programador es responsable del correcto formato del código, evitando errores tales como el mal uso de las etiquetas o el orden incorrecto de los campos de las instrucciones.

Se muestra, para terminar, el proceso de tres etapas que convierte el código inicial en el código máquina entendible por la CPU; eliminación de comentarios, sustitución de etiquetas y traducción.