Organización y Arquitectura de Computadoras
2015-2
Práctica 8

Profesor: José de Jesus Galaviz Casas
Ayud. lab.: Roberto Monroy Argumedo

12 de mayo del 2015
Fecha de entrega: 19 de mayo del 2015

Escribe en el lenguaje de programación C un ensamblador para el lenguaje ensamblador de DLX descrito en las notas del curso. El lenguaje será usado en la siguiente práctica en la que se elaborará un simulador del datapath de DLX.

1. Entrada

Un archivo de texto con las instrucciones de un programa codificado en lenguaje ensamblador. El nombre de archivo se pasará al programa por la linea de comandos.
Ejemplo:

```
./a.out ejemplo.asm
```

2. Salida

Un archivo de texto con las instrucciones codificadas en binario, el texto serán cadenas de carcarteres '0' y '1', una instrucción por línea. El nombre del archivo de salida será el mismo que el archivo de entrada pero con extensión .o.

3. El lenguaje

1. Solo se permite una instrucción por línea y no se permiten líneas en blanco.

2. Los argumentos de las instrucciones estarán separados por espacios o tabulaciones.

3. Se contará con 32 registros de propósito general. El nombre de los registros comenzará con el símbolo $ seguido de dos dígitos indicando el número del registro. Ejemplo: El registro 2 se codificará como `$02`. El registro `$00` será constante con valor 0.

4. Se permitirá el uso de etiquetas para las direcciones de la memoria de instrucciones. La etiqueta sólo puede contener carcarteres alfabéticos finalizando con dos puntos (:).
   Ejemplo:

   ```
   label: add $01 $01 $01
   ```

5. El único modo de direccionamento para la memoria de datos será por desplazamiento. Ejemplo: Suponiendo que se quiere cargar la palabra guardada en la dirección de memoria `0x0002`:

   ```
   lw $01 2($00)
   ```

6. El tamaño de cada instrucción será de 32 bits, se seguirán los formatos de instrucción definidos en las notas del curso. Deberás asignar los opcodes como creas conveniente, recuerda que serán usados para decodificar las instrucciones en la siguiente práctica.

7. El lenguaje constará de las siguientes instrucciones:

   - `lw rs imm(rt)`
     Carga una palabra guardada en la dirección `imm + rt` de la memoria de datos en el registro `rs`.
     Ejemplo:
     `lw $01 0($00)`
   - `lh rs imm(rt)`
     Carga una media palabra guardada en la dirección `imm + rt` de la memoria de datos en el registro `rs`.
     Ejemplo:
     `lh $01 0($00)`

- `lb rs imm(rt)`
  Carga un byte guardado en la dirección `imm + rt` de la memoria de datos en el registro `rs`.
  Ejemplo:
  `lb $01 0($00)`
- `sw ra imm(rt)`
  Guarda una palabra en la dirección `imm + rt` de la memoria de datos de el registro `rs`.
  Ejemplo:
  `lw $01 0($00)`
- `sh rs imm(rt)`
  Guarda una media palabra en la dirección `imm + r` de la memoria de datos de el registro `rt`.
  Ejemplo:
  `lh $01 0($00)`
- `sb rs imm(rt)`
  Guarda un byte en la dirección `imm + rt` de la memoria de datos de el registro `rs`.
  Ejemplo:
  `lb $01 0($00)`
- `add rd rs rt`
  Suma el contenido del registro `rs` y `rt`, guardando el resultado en el registro `rd`.
  Ejemplo:
  `add $01 $01 imm`
- `addi rd rs imm`
  Suma el contenido del registro `ra` y el valor constante `imm`, guardando el resultado en el registro `rd`.
  Ejemplo:
  `add $01 $01 42`
- `sub rd rs rt`
  Resta el contenido del registro `rt` al registro `rs`, guardando el resultado en el registro `rd`.
  Ejemplo:
  `sub $01 $01 $01`
- `subi rd rs imm`
  Resta la constante `imm` al registro `rs`, guardando el resultado en el registro `rd`.

Ejemplo:
`add $01 $01 42`

- `and rd rs rt`
  Realiza la operación lógia and bit a bit entre lo registros `rs` y `rt`, guardando el resultado en `rd`.
  Ejemplo:
  `and $01 $02 $03`
- `andi rd rs rt`
  Realiza la operación lógia and bit a bit entre el registro `rs` y la constante `imm`, guardando el resultado en `rd`.
  Ejemplo:
  `andi $01 $02 31`
- `or rd rs rt`
  Realiza la operación lógia or bit a bit entre lo registros `rs` y `rt`, guardando el resultado en `rd`.
  Ejemplo:
  `or $01 $02 $03`
- `ori rd rs rt`
  Realiza la operación lógia or bit a bit entre el registro `rs` y la constante `imm`, guardando el resultado en `rd`.
  Ejemplo:
  `orii $01 $02 31`
- `beq rs rt label`
  Salto condicional, se ejecuta la instrucción marcada por `label` si `rs` es igual a `rt`.
  Ejemplo:
  `beq rs rt label`
- `bgt rs rt label`
  Salto condicional, se ejecuta la instrucción marcada por `label` si `rs` es mayor a `rt`.
  Ejemplo:
  `bgt rs rt label`
- `j label`
  Salto, se ejecuta la instrucción marcada por `label`.
  Ejemplo:
  `j label`
- `jr rd`
  Salto, se ejecuta la instrucción con la dirección guardada en el

registro `rd`.
Ejemplo:
`jr $01`

4. Entrega

- Se deberá entregar en un archivo `tar` los archivos de código fuente y un documendo PDF en donde se especifique el nombre completo de la persona que realizó la práctica, los opcodes de cada instrucción e instrucciones de compilación y ejecución del programa.
- El programa debe estar completamente documentado y se deben seguir las convenciones vistas en clase (resaltando que no se debe usar break, continue ni etiquetas).
- De no seguir los lineamientos se restarán puntos a la calificación final.