Advanced Reconfigurable Computing DLX Assembler Requirements

Introduction

Your task is to design an assembler that translates DLX assembly language programs into Intel .mif files (one file for data segment and one for code segment).

Requirements

1. You can use any programming language you prefer to write your assembler. Suggested languages are C, C++, or Python. Write good, maintainable code as we are going to use the assembler throughout the semester.

2. The assembler should handle the entire DLX instruction set described in the class lecture notes. Remember that we are implementing a super-sub-set of DLX, so be careful when looking at 3rd party references. The assembler should also be able to identify code and data segments, variables (remember all variables are arrays), and labels.

3. Executing the compiler from the command line should follow this format:

   ```
   $ ./dlx_asm <source_file>.dlx <data_file>.mif <code_file>.mif
   ```

4. Embedding comments in the .mif files may be useful, but is not required.

Pass-off

This is a complex assignment, thus there is a two-part pass-off.

1. 10 points – Demonstrate that your tool can successfully ingest example1.dlx, example2.dlx, and example3.dlx, and produce the appropriate output files for each input.

2. 10 points – Write your own DLX program, using at least one function call, that computes a factorial. You should have two scalar variables, one named `n` that contains the original number and one named `f`, in which the result of the factorial computation is stored. Demonstrate your source code and your successful .mif outputs.

The following pages contain the source code (.dlx) for the three examples plus data segment and code segment outputs for each.

example1.dlx

```
;This is a comment
;Comments always consume entire lines

.data

n          1     10
result     1     0

.text
      ADDI  R1, R0, 0
      ADDI  R2, R0, 1
      LW    R10, n(R0)

top
      SLEI  R11, R10, 1
      BNEZ  R11, exit
      ADD   R3, R1, R2
      ADDI  R1, R2, 0
      ADDI  R2, R3, 0
      SUBI  R10, R10, 1
      J     top

exit
      SW    result(R0), R3

done
      J     done
```

data1.mif

```
DEPTH = 1024;
WIDTH = 32;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

000 : 0000000A;  --n[0]
001 : 00000000;  --result[0]

END;
```

code1.mif

```
DEPTH = 1024;
WIDTH = 32;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

000 : 10200000; --ADDI     R1, R0, 0
001 : 10400001; --ADDI     R2, R0, 1
002 : 05400000; --LW       R10, n(R0)
003 : 816A0001; --SLEI     R11, R10, 1
004 : B160000A; --BNEZ     R11, 00A
005 : 0C611000; --ADD      R3, R1, R2
006 : 10220000; --ADDI     R1, R2, 0
007 : 10430000; --ADDI     R2, R3, 0
008 : 214A0001; --SUBI     R10, R10, 1
009 : B4000003; --J        003
00A : 08600001; --SW       result(R0), R3
00B : B400000B; --J        00B

END;
```

example2.dlx

```
.data

m     1     10
n     1     5
a     1     0

.text

      LW    R5, m(R0)
      LW    R6, n(R0)
      ADDI  R1, R5, 0
      ADDI  R2, R6, 0
      JAL   func
      SW    a(R0), R1

done
      J     done

func
      ;R7 is loop counter
      ;R8 is sum
      ADDI  R7, R0, 0
      ADDI  R8, R0, 0

loop
      ADDI  R9, R0, 0
      SLTI  R9, R7, 32
      BEQZ  R9, loop_done
      ANDI  R10, R2, 1
      BEQZ  R10, if_done
      ADDI  R8, R8, 1

if_done
      SLLI  R1, R1, 1
      SRLI  R2, R2, 1
      J     loop

loop_done
      JR    R31
```

data2.mif

```
DEPTH = 1024;
WIDTH = 32;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

000 : 0000000A;  --m[0]
001 : 00000005;  --n[0]
002 : 00000000;  --a[0]

END;
```

code2.mif

```
DEPTH = 1024;
WIDTH = 32;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

000 : 04A00000; --LW      R5, m(R0)
001 : 04C00001; --LW      R6, n(R0)
002 : 10250000; --ADDI    R1, R5, 0
003 : 10460000; --ADDI    R2, R6, 0
004 : BC000007; --JAL     007
005 : 08200002; --SW      a(R0), R1
006 : B4000006; --J       006
007 : 10E00000; --ADDI    R7, R0, 0
008 : 11000000; --ADDI    R8, R0, 0
009 : 11200000; --ADDI    R9, R0, 0
00A : 61270020; --SLTI    R9, R7, 32
00B : AD200012; --BEQZ    R9, 012
00C : 31420001; --ANDI    R10, R2, 1
00D : AD40000F; --BEQZ    R10, 00F
00E : 11080001; --ADDI    R8, R8, 1
00F : 48210001; --SLLI    R1, R1, 1
010 : 50420001; --SRLI    R2, R2, 1
011 : B4000009; --J       009
012 : B800001F; --JR      R31

END;
```

example3.dlx

```
.data

m     10    1 2 3 4 5 6 7 8 9 10
a     10    0 0 0 0 0 0 0 0 0 0

.text

      ;R12 is loop counter
      ADD   R12, R0, R0

m_loop
      LW    R5, m(R12)
      ADD   R6, R5, R0
      ADD   R1, R5, R0
      ADD   R2, R6, R0
      JAL   func
      SW    a(R12), R1
      ADDI  R12, R12, 1
      SGEI  R13, R12, 10
      BEQZ  R13, m_loop

done
      J     done

func
      ;R7 is loop counter
      ;R8 is sum
      ADDI  R7, R0, 0
      ADDI  R8, R0, 0

f_loop
      ADDI  R9, R0, 0
      SLTI  R9, R7, 32
      BEQZ  R9, loop_done
      ANDI  R10, R2, 1
      BEQZ  R10, if_done
      ADDI  R8, R8, 1

if_done
      SLLI  R1, R1, 1
      SRLI  R2, R2, 1
      J     f_loop

loop_done
      JR    R31
```

data3.mif

```
DEPTH = 1024;
WIDTH = 32;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

000 : 00000001;  --m[0]
001 : 00000002;  --m[1]
002 : 00000003;  --m[2]
003 : 00000004;  --m[3]
004 : 00000005;  --m[4]
005 : 00000006;  --m[5]
006 : 00000007;  --m[6]
007 : 00000008;  --m[7]
008 : 00000009;  --m[8]
009 : 0000000A;  --m[9]
00A : 00000000;  --a[0]
00B : 00000000;  --a[1]
00C : 00000000;  --a[2]
00D : 00000000;  --a[3]
00E : 00000000;  --a[4]
00F : 00000000;  --a[5]
010 : 00000000;  --a[6]
011 : 00000000;  --a[7]
012 : 00000000;  --a[8]
013 : 00000000;  --a[9]

END;
```

code3.mif

```
DEPTH = 1024;
WIDTH = 32;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

000 : 0D800000; --ADD      R12, R0, R0
001 : 04AC0000; --LW       R5, m(R12)
002 : 0CC50000; --ADD      R6, R5, R0
003 : 0C250000; --ADD      R1, R5, R0
004 : 0C460000; --ADD      R2, R6, R0
005 : BC00000B; --JAL      00B
006 : 082C000A; --SW       a(R12), R1
007 : 118C0001; --ADDI     R12, R12, 1
008 : 91AC000A; --SGEI     R13, R12, 10
009 : ADA00001; --BEQZ     R13, 001
00A : B400000A; --J        00A
00B : 10E00000; --ADDI     R7, R0, 0
00C : 11000000; --ADDI     R8, R0, 0
00D : 11200000; --ADDI     R9, R0, 0
00E : 61270020; --SLTI     R9, R7, 32
00F : AD200016; --BEQZ     R9, 016
010 : 31420001; --ANDI     R10, R2, 1
011 : AD400013; --BEQZ     R10, 013
012 : 11080001; --ADDI     R8, R8, 1
013 : 48210001; --SLLI     R1, R1, 1
014 : 50420001; --SRLI     R2, R2, 1
015 : B400000D; --J        00D
016 : B800001F; --JR       R31

END;
```