CS 245 Fall 2018

Assignment 2 – Parsing a RISC-V Instruction

For all assignments, be sure to:

- Include file comments at the top, listing your name, the name of the program, and briefly describe what the program does.
- Describe the register convention, and calling sequence for procedures.
- Line up instruction parts in columns: 0=labels; 1tab=instruction mnemonic; 2tabs=operands; 3-4tabs=comments.
- Include pseudo-code comments to the right of your assembly code. **Avoid** comments that tell me what the assembly does, such as: move 25 to register $s3 – I already know this. Keep your comments to the logic.

Program 2: Parsing a RISC-V instruction

In this assignment, you will take **a RISC-V instruction stored in memory**, and print it in normal printed format.

RISC-V uses a number of formats for its instructions. The first format is for an R-type instruction, with three registers. The second type is the S-type instruction, which is for Store instructions. The third row shows the bit allocations. The fourth row is for the I-type instructions, for use with immediates. See the example below.

R-Format	Func7	RS2	RS1	Func3	RD	Opcode
S-Format	Imm[11:5]	RS2	RS1	Func3	Imm[4:0]	Opcode
Bit alloc.	31..25	24..20	19..15	14..12	11..7	6..0
I-Format	Imm[11:0]		RS1	Func3	RD	Opcode

Here are some example instructions, with their machine code equivalents. Notice that instead of having $t and $s registers, they are simply referred to as x registers. The opcodes provided in the table are appropriate for a number of I, R and S type instructions. Thus, you can tell the format of the instruction from the provided opcodes.

Instruction	Meaning	Machine Code	Opcodes	Output
add x1,x2,x3	Add	0x003100b3	0110011	R-format x1,x2,x3
ld x1,1000(x2)	Load doubleword	0x3e813083	0010011 or 0000011	I-format x1,x2,1000
sd x1,1000(x2)	Store doubleword	0x3e113423	0100011	S-format x1,x2,1000

Your job in this homework is to take the machine code listed above as input and print out the Output column above. This means parsing the instruction, determining its format from the

opcode, and printing the output as shown. Print the instruction type, as well as the registers or immediate value.

To get your logic correct, why not write this in java or pseudocode then write the assembly for it?

Submission

Turn this program in as homework2.asm **via paper and electronic copy to Canvas**. I will write comments on the paper version but can only give you a grade online if you submit to Canvas too.

Grading

Each homework assignment is worth 10 points. Be careful to include all comments and format correctly, as directed in the beginning of this file.

Assignment 3: Using Procedures

In assignment 3 you will enhance your assignment 2 to include the following functions:

- Process coded instructions in a loop: Create an array of hard-coded instructions and process through them one by one. Stop when you load an instruction=0.
- Identify unrecognized opcode and print error message.
- Dynamically determine all fields (registers, function codes and immediates) and print result in formatted way. I will dynamically allocate my own instructions that will likely not be the instructions listed above.
- Print additional fields out: Func3 and Func7.
- Use procedures to handle repeat code, using stacks and arguments passed on stack or in $a registers. Be sure to provide full comments (see top of page) for each procedure. I recommend taking my procedure comments below and using them directly.

I am looking for at least one procedure where you pass parameters on the stack, and one procedure where you pass parameters in the $a registers. Use the stack to store return addresses and saved $s registers.

Print additional Fields

Notice the enhanced output, including the Func3 and Func7 output, printed in hexadecimal.

Instruction	Meaning	Machine Code	Opcodes	Output
add x1,x2,x3	Add	0x003100b3	0110011	R-format x1,x2,x3 Func3=0xN Func7=0xNN
ld x1,1000(x2)	Load doubleword	0x3e813083	0010011 or 0000011	I-format x1,x2,1000 Func3=0xN
sd x1,1000(x2)	Store doubleword	0x3e113423	0100011	S-format x1,x2,1000 Func3=0xN

Use of Procedures

Part of this assignment is to use procedures properly. Create procedures to:

- Decode_Instruction(int instruction): Saves off into a Parsed table the opcode, the source register number, the destination register number and the immediate value. These fields can be accessed any time by any routine.
- Print_Reg_Num(int reg_num): Prints a 'x' and a register number. Used when printing the instruction.
- Print_Format(int opcode): Interprets the opcode passed as a parameter and prints the type: I-Format, R-Format or S-Format. Returns a code in $v0 indicating the format type.

In main(), you may want to do one big switch-case statement: case of R-Format, I-Format and S-Format. In each case, print the information about the instruction.

The previous program was excessively long. So create procedures to help with the printing. Here are the procedures, with their MANDATORY calling sequence:

```
##########################################
#  Decode_Instruction(short instruction):  Saves off into a Parsed table the opcode, the source
#  register number, the destination register number and the immediate value.  These fields can be
#  accessed any time by any routine.
#  Calling Sequence:
#               lw      $a0,instruct    # instruction to parse
#               la      $a1,Parsed      # address of array to parse to
#               jal     Decode
#
#  Returns: A completed Parsed array  (No return parameters)
#
#  Stack Usage:  (You specify, minimally include:)
#                       Return Address
##########################################

##########################################
#  Print_Reg_Num(int reg_num):  Prints a 'x', a register number, and a space.  Used when
#  printing the instruction.
#  Calling Sequence:
#               lw      $a0,reg_num
#               jal     PrReg
#
#  Returns: nothing; Output: The routine prints:  xn
#
#  Stack Usage:  (You specify, minimally include:)
#                       Return Address
##########################################

##########################################
# Print_Format(int opcode):  Interpret the format of the instruction.  Print the format type.
#
# Calling Sequence:  (on the stack)
```

```
#               lw      $a0,opcode
#               sw      $a0, -4($sp)
#               jal     PrtFormat
#
# Returns: $v0=Format type as a code
#
# Stack Usage: (You specify)
##############################################
```

You will be graded on proper use of procedures, stack and documentation. Be sure to include storage of the return address on the stack, as well as any $s registers you use. Please follow the suggested documentation and be sure to document calling sequence, registers conventions, and stack usage for each procedure. Please copy my documentation and adjust it as necessary.

Grading

Each homework assignment is worth 10 points. Be careful to include all comments and format correctly, as directed in the beginning of this file. Please turn in a paper copy to me in class by 5 PM and also submit to Canvas.