ECEC 355
Drexel University, Winter 2021
2/5/2021

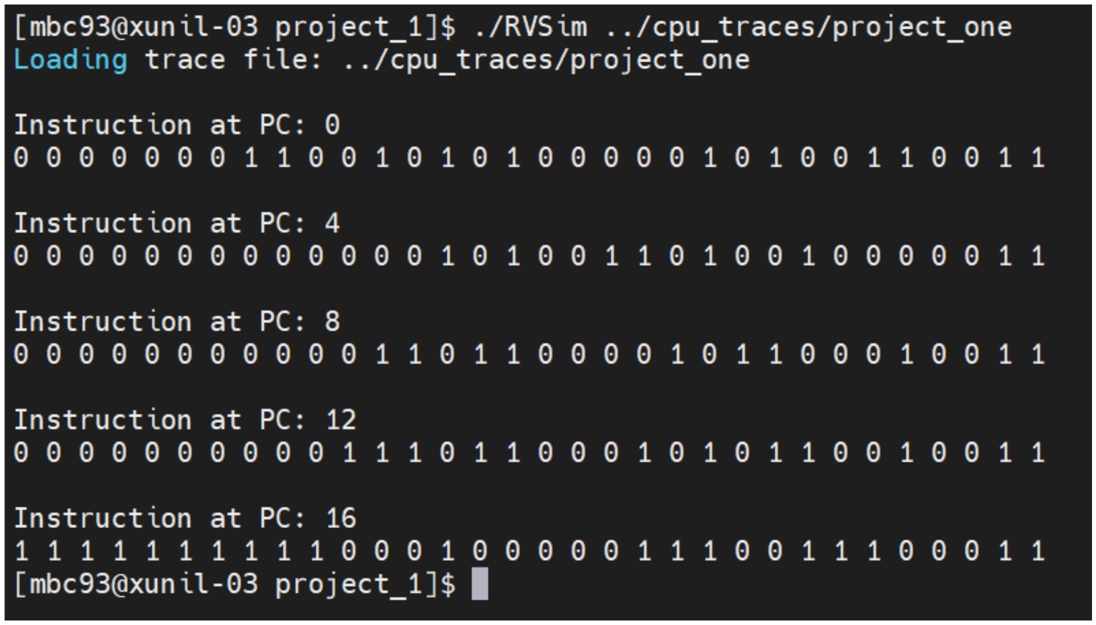
David Schmidt
Mark Carson

Project #1

Procedure Explanation:

1. **Translate Instructions By Hand:** Assembly instructions were translated manually using the textbook, lecture slides and greencard. Manual translations are given following the procedure explanation with a breakdown of each field and the machine instructions as translated and verified by program output.

2. **Edit Parser.c to Support Translation of Given Assembly Instructions:** Parser.c for project_1, as given in the github ECEC355 directory, was extended to add support for the given assembly commands. R-Type commands were supported in the started code. The code was extended to include support for I-Type, and B-Type (SB-Type). The code was extended such that it supported the commands ld, addi, slli, and bne in addition to the support for some R-Type commands that was given in the starter code. The most challenging parts were parsing the base-plus-offset format of the destination register in the ld command and parsing the requisite fields for the bne command. A screenshot of the generated output is given below.



```
[mbc93@xunil-03 project_1]$ ./RVSim ../cpu_traces/project_one
Loading trace file: ../cpu_traces/project_one

Instruction at PC: 0
0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1

Instruction at PC: 4
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1

Instruction at PC: 8
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1

Instruction at PC: 12
0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 1

Instruction at PC: 16
1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1
[mbc93@xunil-03 project_1]$
```

c355 Project_1 manual translations
==================================

add x10, x10, x25
ld x9 (0x10)
addi x22, x22, 1
slli x11, x22, 3
bne x8, x24, -4

----
add x10, x10, x25 (answer given by ./RVsim ../cpu_traces/project_one)

instruction type: R -> funct7 rs2 rs1 funct3 rd opcode
opcode: 0110011
rd: 10 -> 01010
funct3:000
rs1: 10 -> 01010
rs2: 25 -> 11001
funct7: 0000000
immediate:
-> immediate[11:5]:
-> immediate[4:0]:

machine instruction manual translation: 0000000 11001 01010 000 01010 0110011
machine instruction program generated output: 0000000 11001 01010 000 01010 0110011
----
ld x9 0(x10)

instruction type: I -> imm[11:0] rs1 funct3 rd opcode
opcode: 0000011
rd: 9 -> 01001
funct3: 011
rs1: 10 -> 01010
rs2: n/a
funct7: n/a
immediate: 0
-> immediate[11:5]: n/a
-> immediate[4:0]: n/a

machine instruction manual translation: 000000000000 01010 011 01001 0000011
machine instruction program generated output: 000000000000 01010 011 01001 0000011

----
addi x22, x22, 1

instruction type: I -> imm[11:0] rs1 funct3 rd opcode
opcode: 0010011
rd: 22 -> 10110
funct3: 000
rs1: 22 -> 10110
rs2: n/a
funct7: n/a
immediate: 1
-> immediate[11:5]: n/a
-> immediate[4:0]: n/a

```
machine instruction manual translation:           000000000001 10110 000 10110 0010011
machine instruction program generated output: 000000000001 10110 000 10110 0010011
```

----
slli x11, x22, 3

instruction type: I -> imm[11:0] rs1 funct3 rd opcode
opcode: 0010011
rd: 11 -> 01011
funct3: 001
rs1: 22 -> 10110
rs2: n/a
funct7: n/a
immediate: 3
-> immediate[11:5]: n/a
-> immediate[4:0]: n/a

```
machine instruction manual translation:           000000000011 10110 001 01011 0010011
machine instruction program generated output: 000000000011 10110 001 01011 0010011
```

----

bne x8, x24, -4

instruction type: SB ~> imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode

****

*** 13-bit immediate with LSB ALWAYS==0 and therefore discarded ***

*** imm[12]-imm[10:5] rs2 rs1 funct3 imm[4:1]-imm[11] opcode   ***

****

opcode: 1100011

rd: n/a

funct3: 001

rs1: 8 -> 01000

rs2: 24 -> 11000

funct7: n/a

immediate (needs to be 13 bits): -4 -> 1111111111100

```
|12| |11| |10:5|     |4:1|   |0|
 1     1    111111    1110   0

|12|  |10:5|     |4:1|  |11|   |0|
 1      111111   1110     1       0
```

machine instruction manual translation: 1 111111 11000 01000 001 1110 1 1100011

machine instruction program generated output: 1 111111 11100 01000 001 1100 1 1100011