COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE HW 2B PART 1 - Adil Hydari

March 19, 2024

Problem 1. Suppose the data stored in this memory are recycled above PC = 28, i.e., 32 stores 0180FACE, 36 stores 01BAC789 and so on. Also, assume that the low memory (byte) address is to the left of your memory map. A, B are two integers arrays. Each element is of size 4 bytes. The base address of A and B are in register x10 and x11, respectively. Assume that variables i and j are in x12, and x13 respectively and also that the machine is Big Endian.
Part 1 (4 pts): What is the RISC-V assembly code for the following C statement?
B[2*i-4] = A[3 *j-i] + 18;
Part 2 (8 pts): Let's further assume that we store the new contents of B[2*i-4] to register x29, assuming now that i=7 and j=4. Also, x10 holds 12 and x11 holds 20. All other registers are initialized to 0x00. . . 000. Then, we execute the instructions in the following snippet of code:

```
sh x29, 7(x10) Memory Address : 0xFF00...00
lw x13, 5(x10) Memory Address : 0xFF00...04
lui x14, 0x81181 //hexadecimal
Retry: ori x14, x14, 32 //decimal
blt x13, x14, Exit
srli x14, x14, 3
jal x15, Retry
and x13, x29, x14
jalr x16,0(x13)
Exit: ....
```

Q1) What are the contents of register x29? Which bytes of memory are affected? Show the results in the illustration.
Q2) What are the final contents of registers: x13, x14, x15, x16?
Q3) Without making too many calculations argue what is the maximum and what is the minimum total instructions required for the program shown in this snippet of code to complete (note that the first instruction is: sh x29, 7(x10))?
Solution:
Part 1:

```
slli x14, x12, 1 ; 2*i
addi x14, x14, -4 ; 2*i - 4
slli x14, x14, 2 ; Convert index to byte offset for B
add x14, x14, x11 ; Address for B[2*i-4]
slli x15, x13, 1 ; 2*j
add x15, x15, x13 ; 3*j
sub x15, x15, x12 ; 3*j - i
slli x15, x15, 2 ; Convert index to byte offset for A
add x15, x15, x10 ; Address for A[3*j-i]
lw x16, 0(x15) ; Load word from A[3*j-i]
addi x16, x16, 18 ; Add 18
sw x16, 0(x14) ; Store it in B[2*i-4]
```

Part 2:
2a:
Address of B[3*j-i] = 12 + (4*3 - 7)*4 = 12 + (12 - 7)*4 = 12 + 5*4 = 12 + 20 = 32 (decimal). At the decimal 32 there is the value: 0x0180FACE. From here we have to consider the store half instruction (sh), which stores x10(12) with an offset of 7 (12+7=19) to x29, we then apply this offset: 0x0180.
2b:
x13: The memory address from which we are loading (5 + x10) is 5 + 12 = 17 (decimal). The content at this address is 0x18926163 (since we are big endian, we go to the next higher-order byte). After the and operation with x29 (0x0180), x13 will be 0x100.

x14: After the first lui and ori, x14 will be 0x81181020 (lui loads the top 20 bits, then ori loads 0x20 into the bottom 12 bits). This value will right-shift every time we pass Retry without branching to Exit.

x15: This register holds the return address from the jal instruction. This value would change with each iteration of the loop and would be the address to jump back to Retry.

x16: Used for the jalr x16,0(x13) instruction, it would hold the return address of the register that follows. Since x13 now contains 0x100, x16 would be set to that address after the jalr.
2c:
If x13 is already greater than or equal to x14 at the first comparison, the branch to Exit will not be taken and the loop will not execute. In this case, the minimum number of instructions executed is simply the number of instructions from the start to the Exit label without looping.
If x13 is less than x14 at every comparison, the loop will execute until x14 is right-shifted enough times to be less than x13. Because x14 is divided by 8 in each iteration, the number of iterations is limited by how many times x14 can be divided by 8 before it becomes less than x13. The exact number of maximum iterations is limited by the number of bits in x14.
Problem 2.
Part 1: (2 pts) Find the shortest sequence of RISC-V instructions that extracts bits 16 down to 11 from register x5 and uses the value of this field to replace bits 31 down to 26 in register x6 without changing the other bits of registers x5 or x6.
Part 2: (2 pts) Find the shortest sequence of RISC-V instructions that adds number 0xC9 C7 C8 B5 59 77 69 88 to register x5. Use only real instructions (not pseudo-instructions).
Part 3: (2 pts) Find the shortest sequence of RISC-V instructions that add number 0x3D FE 18 4D 92 3E AA 63 to register x5. Use only real instructions (not pseudo-instructions).
Q1) (1 pts) Which one of the three parts requires the fewest instructions?
Q2) (BONUS) Explain in your own words why in order to "append" a negative offset through addi (when using the pair lui-addi) we need to add a +1 to the constant handled by instruction lui no matter if the MSB of the lui constant is positive or negative?
Solution:

1a:
Extract bits 16 to 11 from x5: slli x7, x5, 15
srli x7, x7, 26
lui x8, 0xfc0ff

and x6, x6, x8 (AND x6 with x8 to clear bits 31 to 26)
slli x7, x7, 26
or x6, x6, x7 (OR x6 with x7 to set bits 31 to 26)
1b:
ui x6, 0xC9C7C
addi x6, x6, 0x8B5
slli x6, x6, 12
addi x6, x6, 0x597
slli x6, x6, 12
addi x6, x6, 0x769
slli x6, x6, 12
addi x6, x6, 0x88
add x5, x5, x6

1c:
lui x6, 0x3DFE1
addi x6, x6, 184
slli x6, x6, 12
addi x6, x6, 0xD92
slli x6, x6, 12
addi x6, x6, 0x3EA
slli x6, x6, 12
addi x6, x6, 0xAA6
slli x6, x6, 4
addi x6, x6, 3
add x5, x5, x6
2a:
Part 1 requires the fewest instructions.
2b:
when you add 1 to the constant handled by the lui instruction, you need to compensate for the "borrow" with the addition of the negative offset with addi. This adjustment is necessary regardless of whether the MSB of the lui constant is positive or negative because the issue arises from the interaction between the two instructions and how two's complement arithmetic handles negative numbers.