CENG3420 Homework 2

Due: Mar. 8, 2023

Solutions

All solutions should be submitted to the blackboard in the format of **PDF/MS Word**.

Q1 (25%)

```
1:  .global _start

2:  .text
3:  _start:
4:      li a1, 13
5:      jal ra, fibonacci
6:      j exit

# fibonacci(a1)
7:  fibonacci:
8:      addi sp, sp, -12
9:      sw s0, 0(sp)
10:     sw s1, 4(sp)
11:     sw ra, 8(sp)
12:     slti t0, a1, 3
13:     bne t0, zero, return_1
14:     li s0, 0
15:     addi s1, a1, 0
16:     addi a1, s1, -1
17:     jal ra, fibonacci
18:     add s0, s0, a0
19:     addi a1, s1, -2
20:     jal ra, fibonacci
21:     add s0, s0, a0
22:     addi a0, s0, 0
23:     j exit_fib
24: return_1:
25:     li a0, 1
26: exit_fib:
27:     lw s0, 0(sp)
28:     lw s1, 4(sp)
29:     lw ra, 8(sp)
30:     addi sp, sp, 12
31:     jr ra

32: exit:
33:     li a7, 1
34:     ecall
```

The assembly code above calculate the fibonacci number via recursion.

1. Write down the recursive equation that calculates $fibonacci(a_1)$. (5%)
2. Which register is used to store the result? (5%)
3. What result will we get if we replace the 4th line with `li a1, 2`? What if `li a1, 5`? (5%)

4. In this program, is registers `s0` saved by the caller or callee? (5%)
5. Which line(s) are operating the stack pointer? (5%)

A1 These are suggested solutions.

1.

$$fibonacci(a_1) = \begin{cases} 1, \text{if } a_1 <= 2, \\ fibonacci(a_1 - 1) + fibonacci(a_1 - 2), \text{otherwise.} \end{cases} \quad (1)$$

2. a0
3. 1 and 5
4. callee
5. 8th and 30th

Q2 Multiplication (25%)

1. Consider the first version of multiplication hardware in Figure 1. We use 4-bit numbers for simplicity. The Multiplicand register, ALU, and Product register are all 8 bits wide, with only the Multiplier register containing 4 bits. Write down the value of each register for each of the steps for calculating $1111_{\text{two}} \times 1101_{\text{two}}$. (The value in the Multiplicand, Product and Multiplier registers. 1111_{two} is multiplicand and 1101_{two} is multiplier.) (10%)

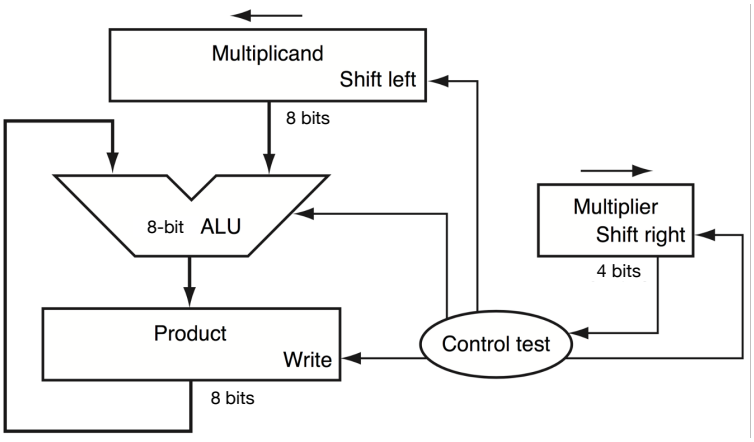


Figure 1: First version of multiplication hardware.

2. Consider the Add and Right Shift Multiplier Hardware in Figure 2. The Multiplicand register and ALU have been reduced to 4 bits. Now the product is shifted right. The separate Multiplier register also disappeared. The multiplier is placed instead in the right half of the Product register, which has grown by one bit to 9 bits to hold the carry-out of the adder. Write down the value of each register for each of the steps for calculating $1111_{\text{two}} \times 1101_{\text{two}}$. (The value in the Product register. 1111_{two} is multiplicand and 1101_{two} is multiplier.) (15%)

A2 These are suggested solutions.

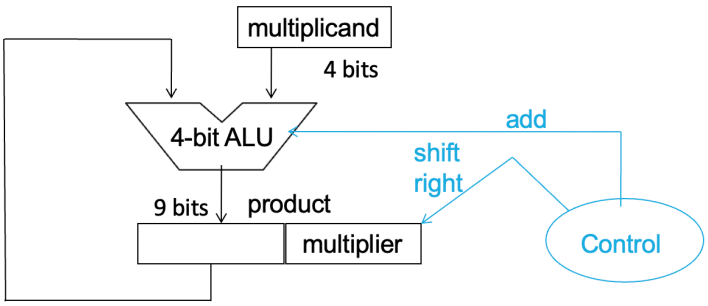


Figure 2: Add and Right Shift Multiplier Hardware.

Table 1: Multiply using simple hardware.

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	1101	0000 1111	0000 0000
1	Prod = Prod + Mcand	1101	0000 1111	0000 1111
	Shift left Multiplicand	1101	0001 1110	0000 1111
	Shift right Multiplier	0110	0001 1110	0000 1111
2	Prod = Prod + Mcand	0110	0001 1110	0000 1111
	Shift left Multiplicand	0110	0011 1100	0000 1111
	Shift right Multiplier	0011	0011 1100	0000 1111
3	Prod = Prod + Mcand	0011	0011 1100	0100 1011
	Shift left Multiplicand	0011	0111 1000	0100 1011
	Shift right Multiplier	0001	0111 1000	0100 1011
4	Prod = Prod + Mcand	0001	0111 1000	1100 0011
	Shift left Multiplicand	0001	1111 0000	1100 0011
	Shift right Multiplier	0000	1111 0000	1100 0011

Table 2: Multiply using add and right shift hardware.

Iteration	Step	Product
0	Initial values	0 0000 1101
1	Prod = Prod + Mcand	0 1111 1101
	Shift right Product	0 0111 1110
2	Prod = Prod + Mcand	0 0111 1110
	Shift right Product	0 0011 1111
3	Prod = Prod + Mcand	1 0010 1111
	Shift right Product	0 1001 0111
4	Prod = Prod + Mcand	1 1000 0111
	Shift right Product	0 1100 0011

1. Table 1 shows the value of each register for each of the steps, with the final value of 11000011_{two} or 195_{ten}.
2. Table 2 shows the value of 9-bit Product register in each step, with the final value of 11000011_{two} or 195_{ten}.

Q3 IEEE 754 Floating-Point Standard (20%)

1. Show the IEEE 754 binary representation of the number $-0.625_{\texttt{ten}}$ in single precision. (Show your work.) (5%)
2. What decimal number does this single precision float $\texttt{3EC00000}_{16}$ represent? (Show your work.) (5%)
3. Please give the range of single precision numbers in IEEE 754. (the range that the normalized numbers can represent in decimal. Show your work.) (10%)

A3 These are suggested solutions.

1. $-0.625 = -5/8 = (-1)\times(5)\times 2^{-3} = (-1)\times(101)_{\texttt{two}}\times 2^{-3} = (-1)\times(1.01)_{\texttt{two}}\times 2^{-1}$. Therefore, the binary representation of $-0.625_{\texttt{ten}}$ is $10111111001000000000000000000000$ or $\texttt{BF200000}_{16}$.
2. The sign bit is 0, and the exponent field is $01111101_{\texttt{two}}$, therefore, the actual exponent is $125-127=-2$. The fraction is 0.5. The decimal number is $1\times(1+0.5)\times 2^{-2} = 0.375$.
3. $1 \le E \le 254, N = (-1)^S \times (1.M) \times 2^{E-127}$. For $S = 0, 2^{-126} \le N \le (1.111\ldots111)\times 2^{127} = (2-2^{-23})\times 2^{127} = 2^{128} - 2^{104}$. For $S = 1$, the analysis is the same. Therefore, $N \in [2^{104} - 2^{128}, -2^{-126}] \cup [2^{-126}, 2^{128} - 2^{104}]$.

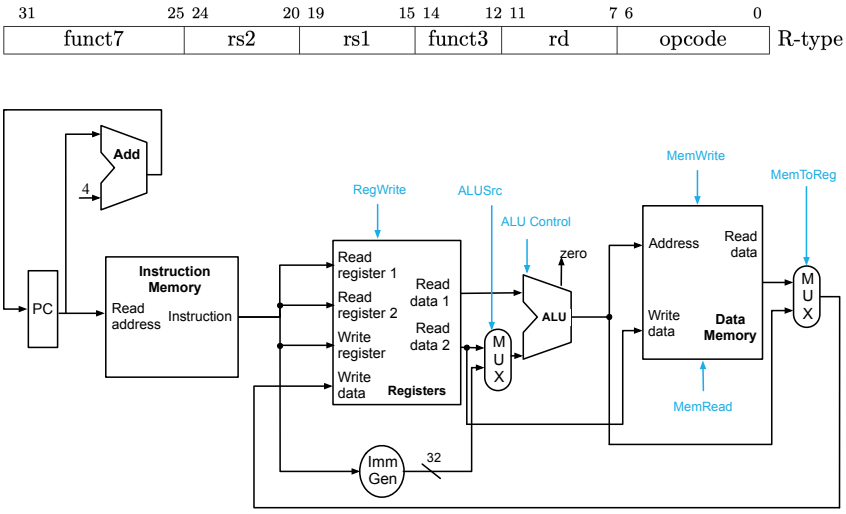
Q4 (10%)

1. What does `PC` stand for? How to get the address of the next instruction? (5%)
2. Write down the three most critical steps in the datapath/control of a processor. (5%)

A4 These are suggested solutions.

1. Program counter. `PC + 4`.
2. Fetch, decode, execute.

Q5 (20%)



These figures show the format and datapath of an R format instruction.

1. Assume we have an instruction whose machine code is `0100000 00010 00001 101 00011 0110011`, please write down the instruction in assembly language. (10%)
2. Which ports in the datapath do we use to specify `rd`, `rs1`, and `rs2`? (5%)
3. What are the functionalities of the following components in the datapath? (a) Add (b) Instruction Memory (c) ALU (5%)

Note that the reference of RISC-V can be found on:

`http://www.cse.cuhk.edu.hk/~byu/CENG3420/2023Spring/doc/RV32-reference-1.pdf`

`http://www.cse.cuhk.edu.hk/~byu/CENG3420/2023Spring/doc/RV32-reference-2.pdf`

A5

1. sra x3, x1, x2
2. Write register, Read register 1, Read register 2
3. (a) PC + 4 (b) Fetch and decode (c) execute