Tutorial 7 – Week of Nov. 1

Question 3.28

The Hewlett-Packard 2114, 2115, and 2116 used a format with the leftmost 16 bits being the fraction stored in two's complement format, followed by another 16-bit field which had the leftmost 8 bits as an extension of the fraction (making the fraction 24 bits long), and the rightmost 8 bits representing the exponent. However, in an interesting twist, the exponent was stored in sign-magnitude format with the sign bit on the far right! Write down the bit pattern to represent -1.5625×10^{-1} assuming this format. No hidden 1 is used. Comment on how the range and accuracy of this 32-bit pattern compares to the single precision IEEE 754 standard.

Solution

3.28 $-1.5625 \times 10^{-1} = -0.15625 \times 10^{0}$

$= -0.00101 \times 2^{0}$

move the binary point two to the right

$= -0.101 \times 2^{-2}$

exponent $= -2$, fraction $= -0.10100000000000000000000000$

answer: 10110000000000000000000000000101

fraction exponent

Question 3.32

IEEE 754-2008 contains a half precision that is only 16 bits wide. The leftmost bit is still the sign bit, the exponent is 5 bits wide and has a bias of 15, and the mantissa is 10 bits long. A hidden 1 is assumed.
Calculate $(3.984375 \times 10^{-1} + 3.4375 \times 10^{-1}) + 1.771 \times 10^{3}$ by hand, assuming each of the values is stored in the 16-bit half precision format. Assume 1 guard, 1 round bit, and 1 sticky bit, and round to the nearest even.

Solution

3.32 $(3.984375 \times 10^{-1} + 3.4375 \times 10^{-1}) + 1.771 \times 10^{3})$

$3.984375 \times 10^{-1} = 1.1001100000 \times 2^{-2}$

$3.4375 \times 10^{-1} = 1.0110000000 \times 2^{-2}$

$1.771 \times 10^{3} = 1771 = 1.1011101011 \times 2^{10}$

shift binary point of smaller left 12 so exponents match

```
(A)          1.1001100000
(B)         +1.0110000000
            -------------
            10.1111100000 Normalize,
(A+B)        1.0111110000 × 2^-1
(C)         +1.1011101011
(A+B)          .0000000000 10 111110000 Guard = 1,
                           Round = 0, Sticky = 1
            ---------------
(A+B)+C     +1.1011101011 10 1 Round up
(A+B)+C = 1.1011101100 × 2^10 = 0110101011101100 = 1772
```

Question: 4.1 rd rs1 rs2

Consider the following instruction:

```
and rd, rs1, rs2
```

Interpretation:

```
Reg[rd] = Reg[rs1] AND Reg[rs2]
```

4.1.1 What are the values of control signals generated by the control in Figure 4.10 for this instruction?

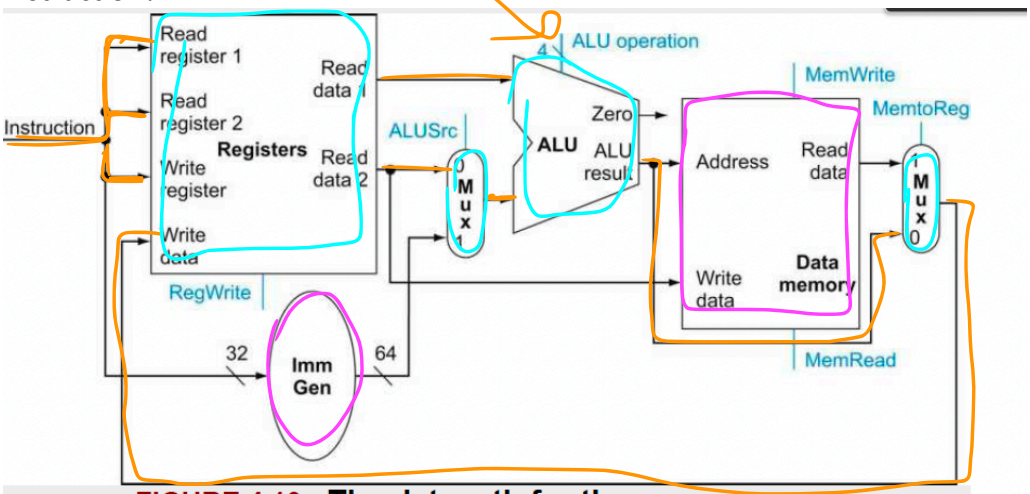


FIGURE 4.10 The datapath for the memory instructions and the R-type instructions.

useful datapath

useful blocks

unuseful blocks

4.1.2 Which resources (blocks) perform a useful function for this instruction?

4.1.3 Which resources (blocks) produce no output for this instruction? Which resources produce output that is not used?

Solution

There is data flowing in all data path, we use MUX to select the data we want and ignore the data we don't want.

So all resources produce some output, although we don't care some of the output.

4.1

4.1.1 The value of the signals is as follows:

RegWrite	ALUSrc	ALUoperation	MemWrite	MemRead	MemToReg
true	0	"and"	false	false	0

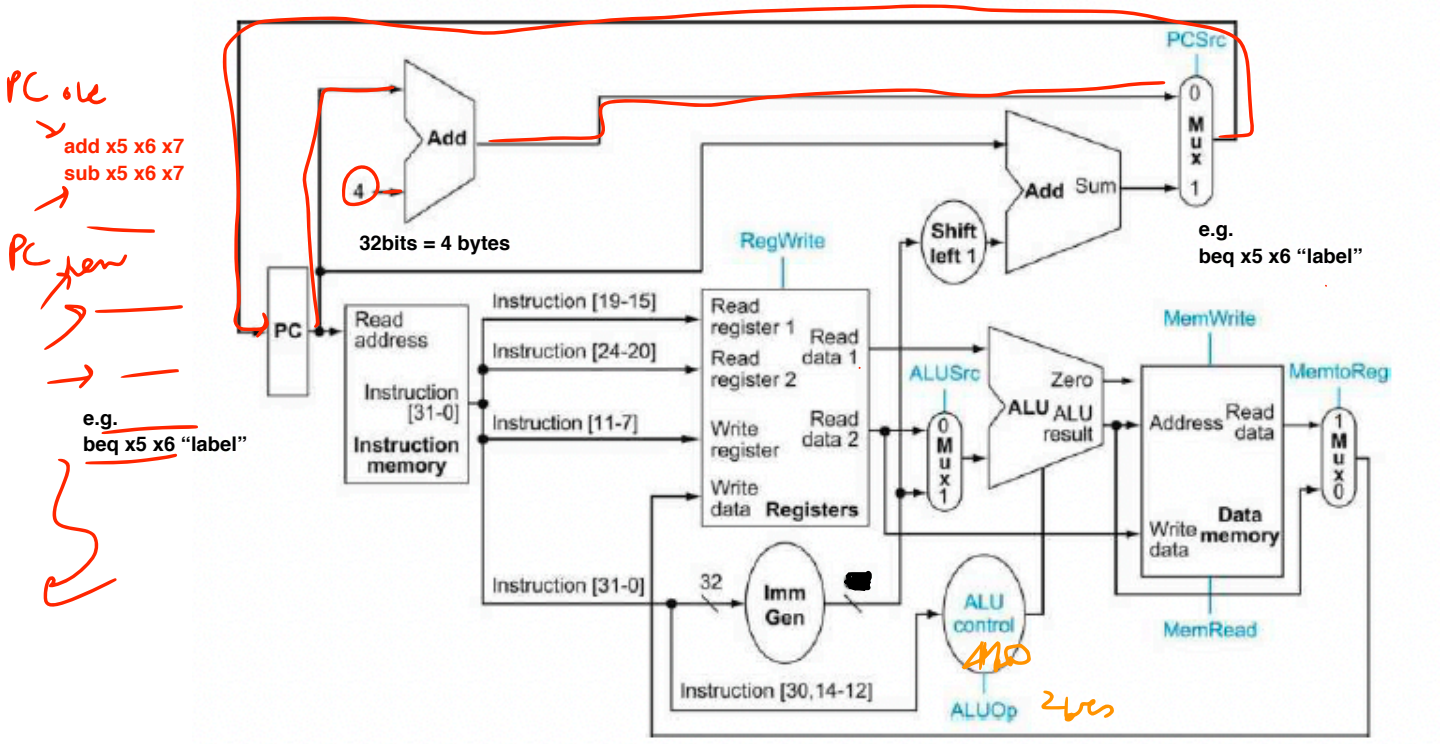
Mathematically, the `MemRead` control wire is a "don't care": the instruction will run correctly regardless of the chosen value. Practically, however, `MemRead` should be set to false to prevent causing a segment fault or cache miss.

4.1.2 Registers, ALUsrc mux, ALU, and the MemToReg mux.

4.1.3 All blocks produce some output. The outputs of DataMemory and Imm Gen are not used.

Question: 4.5

In this exercise, we examine in detail how an instruction is executed in a single-cycle datapath.



Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word:

`0x00c6aa23`

4.5.1 What are the values of the ALU control unit's inputs for this instruction?

4.5.2 What is the new PC address after this instruction is executed?

4.5.3 For each mux, show the values of its inputs and outputs during the execution of this instruction. List values that are register outputs at Reg [xn].

4.5.4 What are the input values for the ALU and the two add units?

4.5.5 What are the values of all inputs for the registers unit?

Solution

4.5

	rs2	imm	rs1
The encoded instruction is	`sw x12, 20(x13)`		

4.5.1 (from Figure 4.12)

`ALUop = 00`

`ALU Control Lines = 0010`

4.5.2
The new PC is the old PC + 4. This signal goes from the PC, through the “PC + 4” adder, through the “branch” mux, and back to the PC.

4.5.3
ALUsrc: Inputs: Reg[x12] and `0x00000014`; Output: `0x00000014`

MemToReg: Inputs: Reg[x13] + `0x14` and <undefined>; output: <undefined>

Branch: Inputs: PC+`4` and PC + `0x28`
Imm = 20(dec) = 10100
shift one bit left
101000 = 40(dec) = 0x28

4.5.4
ALU inputs: Reg[x13] and `0x00000014`
PC + 4 adder inputs: PC and 4
Branch adder inputs: PC and `0x00000028`

4.5.5
Read register 1 = `0x13` (base address)
Read register 2 = `0x12` (data to be stored)
Write register = `0x0` or don’t-care (should not write back)
Write data = `don't-care` (should not write back)
RegWrite = `false` (should not write back)