CSE-340

Assignment-2

Name: MD Ikramul Kayes

ID: 2130157 6

Sec: 04

Ans: to: que: no: 1

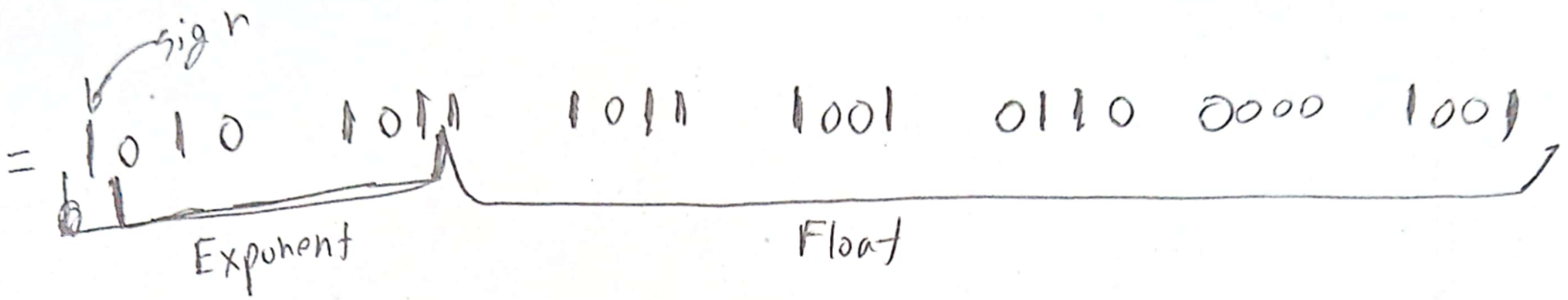
Multiplicand = 13 = 1101

Multiplier = 17 = 10001

Iteration	Multiplicand	Product
0	1101	0000 10001
1	1101	1101 10001 0110 11000
2	1101	0011 01100
3	1101	0001 10110
4	1101	0000 11011
5	1101	1101 11011 0110 11101

Ans: to: que: no 2

A B B 9 6 0 9



Biased Exp = 21

Bias = $2^{6-1} - 1 = 31$

Exponent = 21 - 31 = -10

Fraction $= 2^{-1} + 2^{-2} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-9} + 2^{-11}$

$+ 2^{-12} + 2^{-18} + 2^{-21}$

$= 0.862064$

$\therefore (-1)^1 (1 + 0.862064) \times 2^{-10}$

$= -1.81842 \times 10^{-3}$

Ans: to: que: no: 03

A] 50.7869 , +79.83 −29.58

$= 110010.11001 + 1001111.11010$

$- 11101.10000$

$= 1.10010\,11001 \times 2^5 + 1.00111\,11010$

$\times 2^6 - 1.11011\,0000 \times 2^4$

$= 0.11001\,0 \times 2^6 + 1.00111 \times 2^6$

$- 1.11011\,0 \times 2^4$

$= 10.00000 \times 2^6 - 1.11011 \times 2^4$

~~$= 1.00000 \times 2^7 - 0.000111 \times 2^7$~~

$= 1.00000 \times 2^7 - 0.00111 \times 2^7$

$= 1.00111 \times 2^6$ 0.11001×2^7

$= 1.1110$

$= 1.10010 \times 2^6$

b) $64.2486 * 49.1832$

$= 1000000.0011\underline{1} * 110001.00101$

$= 1.00000 \times 2^6 * 1.10001 \times 2^5$

$= 1.10001 \times 2^{11}$

Ans: to: que: no: 4

$28.4810 - (-4.0210)$

$= 28.4810 + 4.0210$

$= 11100.01111 + 100.000$

$= 1.1100011111 \times 2^4 + 1.00 \times 2^2$

$= 1.1100011111 \times 2^4 + 0.0100 \times 2^4$

$= 10.0000111111 \times 2^4$

$= 1.0000001111 \times 2^5$

Bias $= 2^{8-1} - 1 = 127$

Biased Exponent $= 127 + 5 = 132$

Range $= 1$ to 254

As, $1 < 132 < 254$, We can say result is none

Ans: to: que: no: 5

a] In IEEE 754 bias is added to exponent field, so that it can represent both positive and negative exponent. using unsigned number. this also reduces complexity as we can compare numbers easily as both negative and positive numbers are represented in unsigned. numbers bias

b] In optimized multiplication reduces the number of components that get used in long multiplication. As, we do not need separate multiplier register and we do not need product sized ALU in optimized multiplication. Because of this we need less equipment also. We can do the multiplication where one shift required for each instruction.

Ans: to: que: no: 6

1] fadd.s

(i) Do floating point addition for single-precision

(ii) ~~fadd.s des, src res1, res2~~

fadd.s des, rsrc1, src2

(iii) Here, src 1, src2 have single-precision floating number which gets added and stores in des register

(iv) fadd.s f3, f4, f5

2] fsub.s

(i) Do floating point substruction for single-precision

(ii) fsub.s des, src1, src 2

(iii) Here, value of src1 get substracted by src2. The result will stored in des.

(iv) fsub f4, f3, f2

3] fmul.s

(i) Do floating point multiplication for single-precision

(ii) fmul.s des, src1, src2

(iii) Here values of src1, src2 both single precision floating number will be multiplied and gets ~~stobe~~ stored in des register

(iv) fmul.s f4, f3, f2

4] fdiv.s

(i) Do floating-point division for single precision number.

(ii) fdiv.s des, src1, src2

(iii) Here, values of src1, src2 both single-precision floating number, src1 will get devided by src2 and the result will be stored in des.

(iv) fdiv.s f2, f3, f4

5] fsqrt.s

(i) Performs floating-point square root for single-precision number

(ii) fsqrt.s des, src

(iii) single-precision floating number of src will get square root and the result will get stored in ~~res~~ des.

(iv) fsqrt.s f20, f13

6) feq.d

(i) Compares between two double precision number, if they are equal ~~Jumps to mentioned d Label.~~ or not.

(ii) feq.d des, src1, src2

(iii) if src1 == src2 then it stores 1 to des register otherwise it stores 0.

(iv) feq.d f X1, f2, f3

7] fle.d

(i) compares between two double precision numbers if one is less than, or equal to the other, number.

(ii) fle.d des, src1, src2

(iii) If src1 is less than or equal to src2 then it stores 1 in des, other wise it stores 0;

(iv) fle.d %3, f1, f2

8] flt.s

(i) Checks if one single-point floating number is less than another number,

(ii) flt.s des, src1, src2

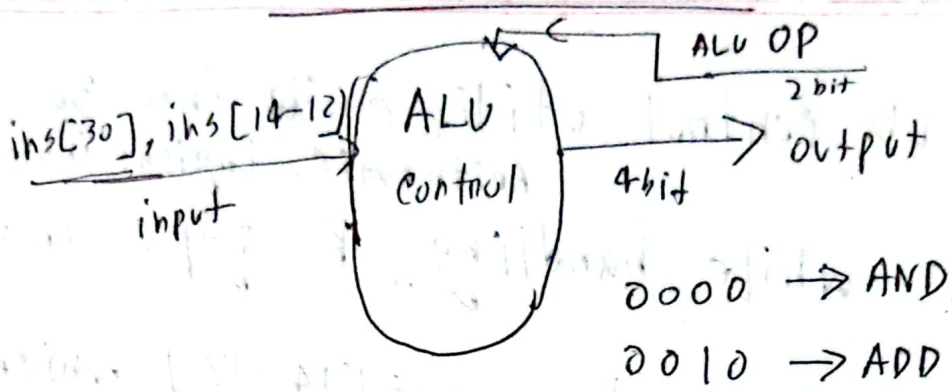
(iii) If src1's value is less than src2 then it stores 1 to des register, otherwise it stores 0 to des register.

(iv) flt.s ~~f~~ X12, f3, f4

Ans to que no 7

```
feq.d  X19, f1, f2
beq  X19, X0, Jump Not Equal
Jump Equal:
 //code for Jump equal
beq X0, X0, exit
Jump Not Equal:
//code for Jump Not Equal
exit:
```

Ans: to: que: no: 8



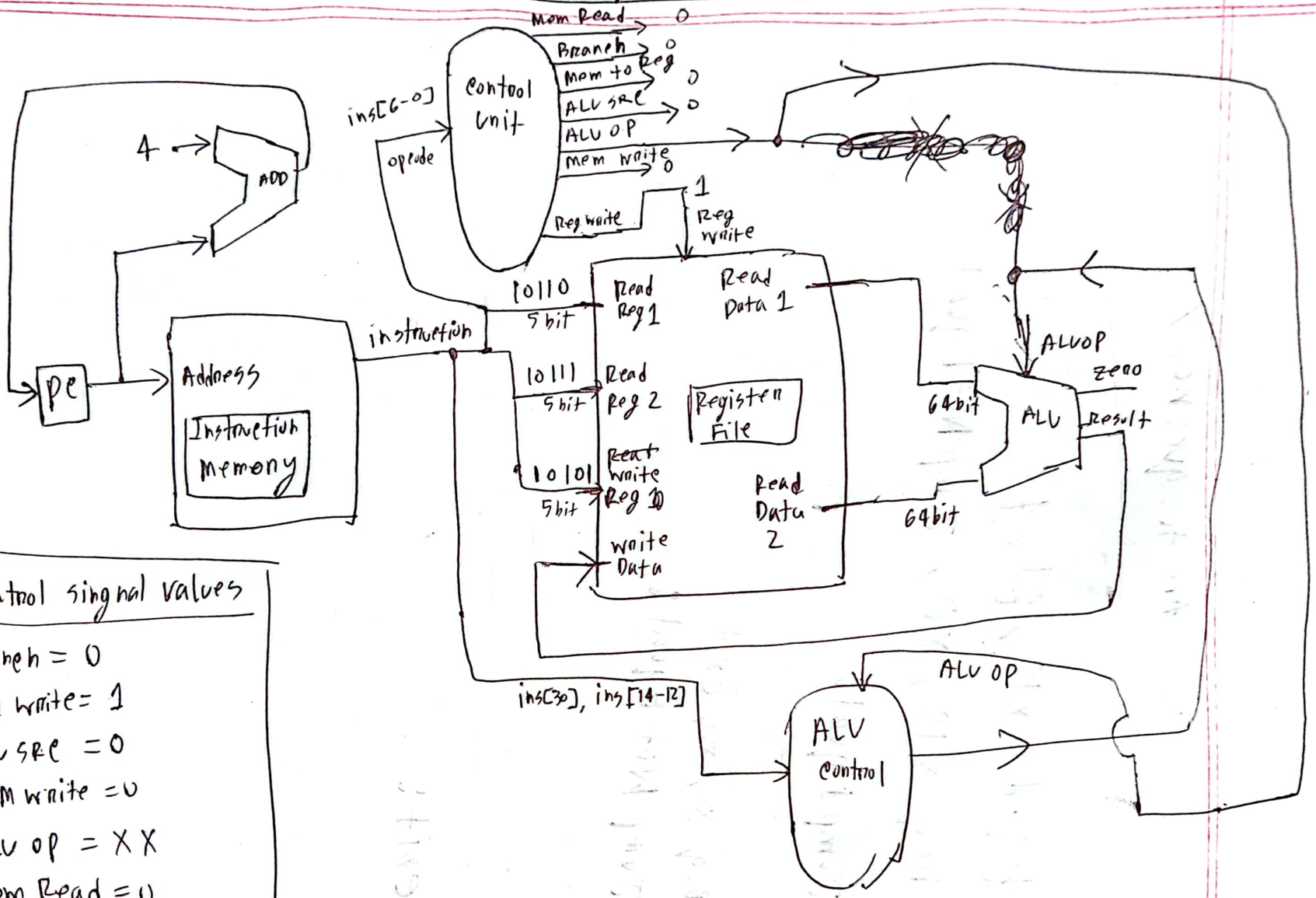
a] Ans: For LD instruction the ALU control does not utilize instruction 30, 14-12 bit to generate output. As for LD operation it do not have to rely on ins[30, 14-12] bits as the ~~Op-code gives ALU, the instruction~~ control unit's ALU op code gives the instruction to ALU control to do. (base + offset) operation. So, it do not need to rely on ins[30, 14-12] ALU op by control unit gives the instruction to load to ALU control unit.

b] ALU control utilize ins 30, 14-12 bits while handling Arithmatic, Logical instructions and R-type instructions.

As in R-type the ins[14-12] which is funct3 let the ALU control decide which instruction of R-type to perform. Based on funct3 it gives ALU to perform ADD/SUB/MUL operation.

On the other hand funct7 ins[30] is used to add more variation in the instruction. some times, funct7 is used as parity checker.

Ans to que no: 9



Control signal values

Branch = 0

Reg write = 1

ALU SRC = 0

MEM write = 0

ALU op = X X

Mem Read = 0

Mem to Reg = 0