Solution CSA (CSL4208)

Section A

Q1. a. (i) RISC

b. (ii) ABC+DE+F*+G/+

c. (ii) Assembly Line operation

d. (ii) Control command

e. (iv)Zero Address Instructions

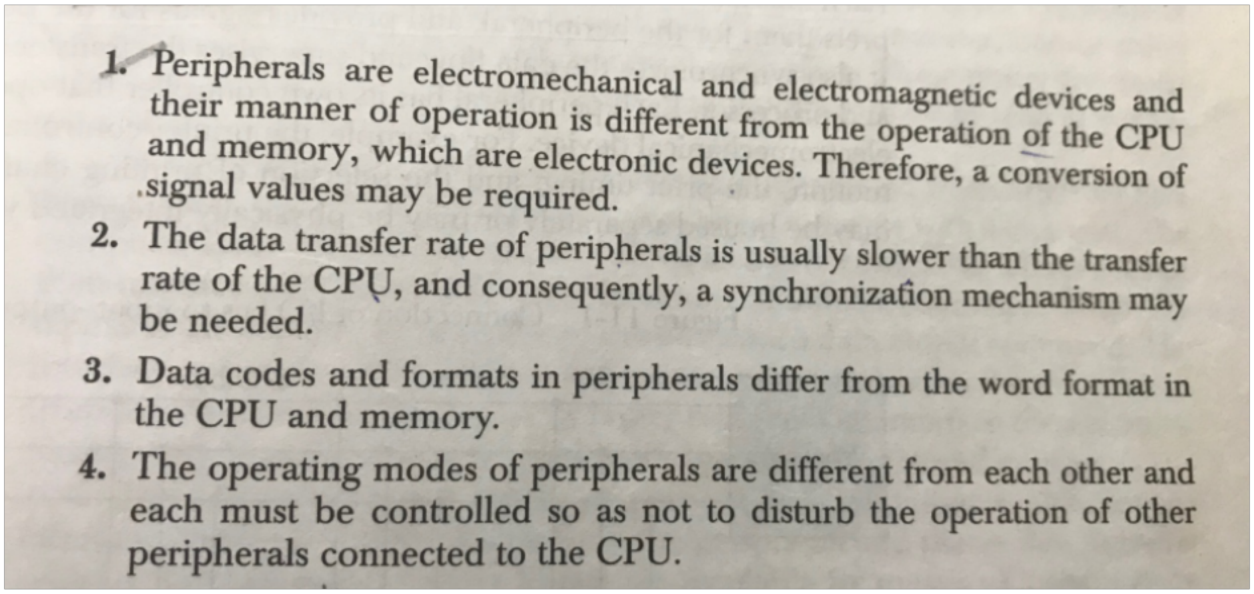
Section B

Q2. i) 010 011 001 00010 ii) 101 xxx 101 00110 or 10100000110

Q3. The address part of the indexed mode instruction must be set to zero. Register Indirect mode: In this addressing the operand's offset is placed in any one of the registers BX,BP,SI,DI as specified in the instruction. The effective address of the data is in the base register or an index register that is specified by the instruction.

Q4.

I/O devices need Interface to communicate with the Common bus. Following are the reasons why they can't communicate directlty with the common bus:

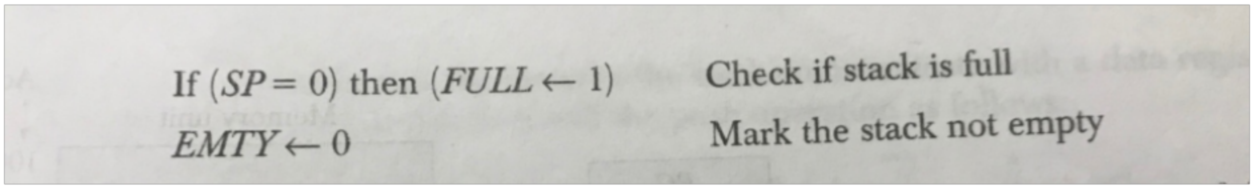


1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.
2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

Q5. i) Status Command

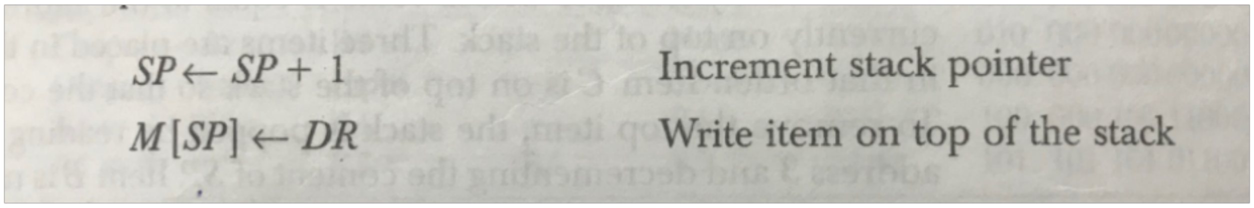
ii) Status Command

Q6. Micro-operations for inserting an element onto the memory stack are as follows:



If $(SP = 0)$ then $(FULL \leftarrow 1)$ Check if stack is full

$EMTY \leftarrow 0$ Mark the stack not empty



$SP \leftarrow SP + 1$ Increment stack pointer

$M[SP] \leftarrow DR$ Write item on top of the stack

Q7. A=240 A= 11110000

B= 20 B= 00010100

A – B= A+2's Complement of B

= 11110000 + 11101100

= $\underline{1}$ 11011100

C=1, V=0

Section C

Q8.

RISC	CISC
1. RISC stands for Reduced Instruction Set Computer.	1. CISC stands for Complex Instruction Set Computer.
2. RISC processors have simple instructions taking about one clock cycle. The average clock cycle per instruction (CPI) is 1.5	2. CSIC processor has complex instructions that take up multiple clocks for execution. The average clock cycle per instruction (CPI) is in the range of 2 and 15.
3. Performance is optimized with more focus on software	3. Performance is optimized with more focus on hardware.
4. It has no memory unit and uses a separate hardware to implement instructions..	4. It has a memory unit to implement complex instructions.
5. It has a hard-wired unit of programming.	5. It has a microprogramming unit.
6. The instruction set is reduced i.e. it has only a few instructions in the instruction set. Many of these instructions are very primitive.	**6.** The instruction set has a variety of different instructions that can be used for complex operations.
7. The instruction set has a variety of different instructions that can be used for	**7.** CISC has many different addressing modes and can thus be used to represent

complex operations.	higher-level programming language statements more efficiently.
8. Complex addressing modes are synthesized using the software.	8. CISC already supports complex addressing modes
9. Multiple register sets are present	9. Only has a single register set
10. RISC processors are highly pipelined	10. They are normally not pipelined or less pipelined
11. The complexity of RISC lies with the compiler that executes the program	11. The complexity lies in the microprogram
12. Execution time is very less	12. Execution time is very high

Q9. i) Minimum clock cycle time=t_p = 75+5= 80ns

ii) Time to perform 100 tasks= $(k+n-1)t_p$

= (4+99)80

=8240ns

iii) Speed Up= nt_n / $(k+n-1)t_p$

t_n = 40+30+75+5= 150

Speed Up= 100X150 / 8240

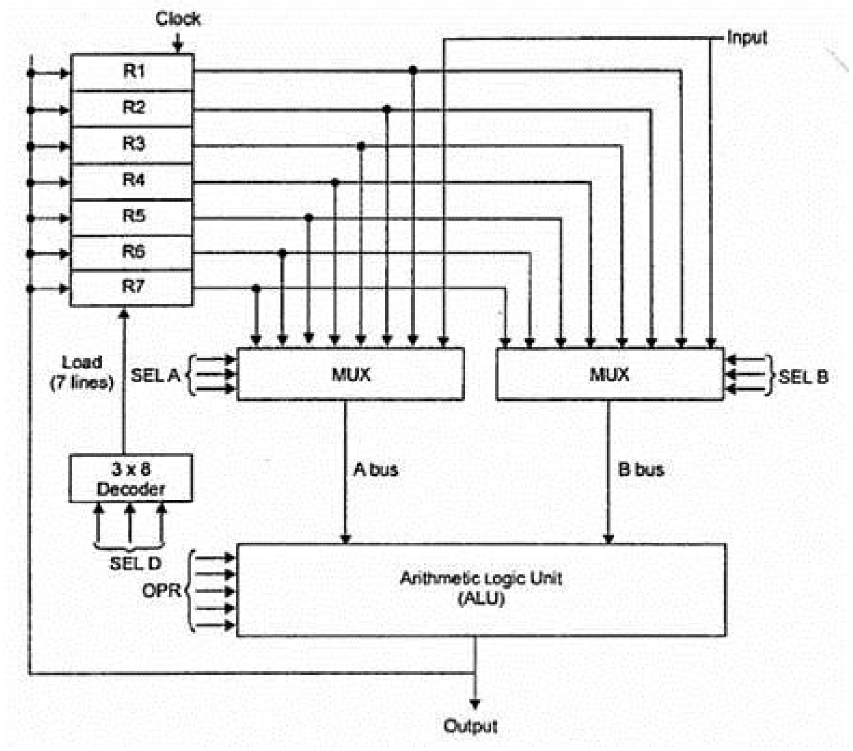
= 1.82

Q10. Generally CPU has seven general registers. Register organization show how registers are selected and how data flow between register and ALU. A decoder is used to select a particular register.The output of each register is connected to two multiplexers to form the two buses A and B. The selection lines in each multiplexer select the input data for the particular bus.

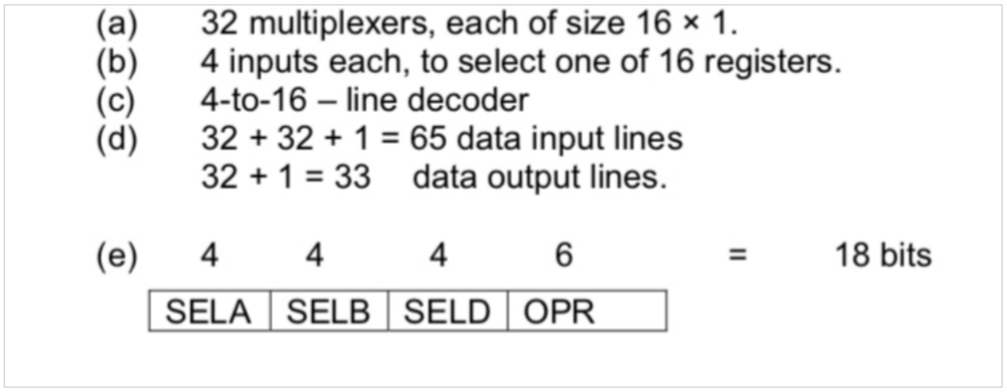
The A and B buses form the two inputs of an ALU.The operation select lines decide the micro operation to be performed by ALU. The result of the micro operation is available at the output bus. The output bus connected to the inputs of all registers, thus by selecting a destination register it is possible to store the result in it.

For Example: To perform the operation ***R3 = R1+R2*** We have to provide following binary selection variable to the select inputs.

1. **SEL A** : **001** -To place the contents of R1 into bus A.
2. **SEL B** : **010** - to place the contents of R2 into bus B
3. **SEL OPR** : **10010** – to perform the arithmetic addition A+B
4. **SEL REG or SEL D : 011** – to place the result available on output bus in R3



Q11.



(a) 32 multiplexers, each of size 16 × 1.

(b) 4 inputs each, to select one of 16 registers.

(c) 4-to-16 – line decoder

(d) 32 + 32 + 1 = 65 data input lines
32 + 1 = 33 data output lines.

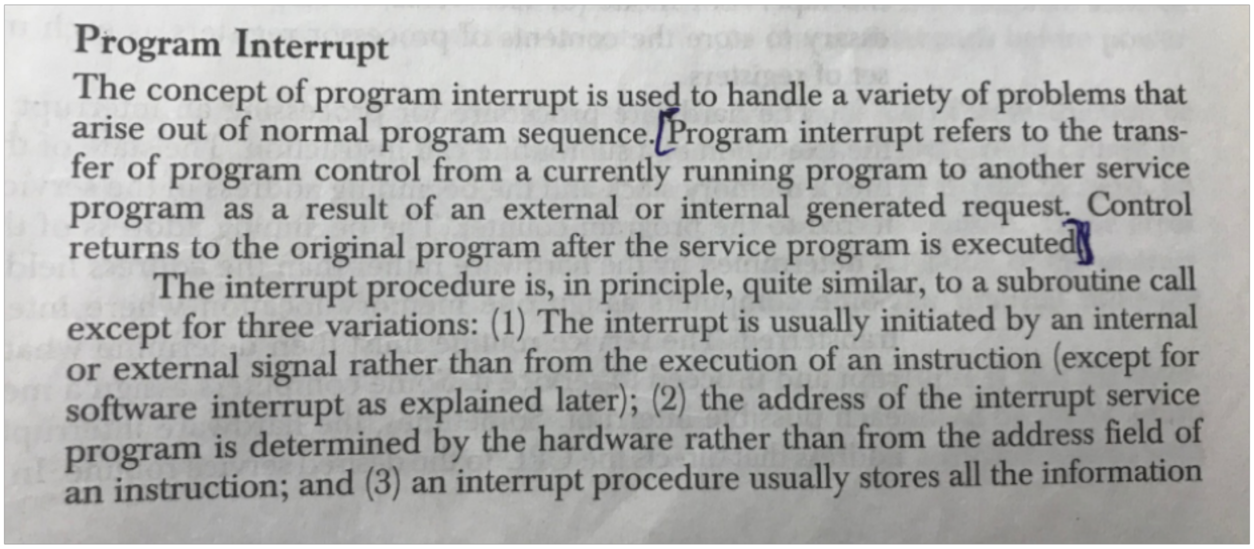
(e) 4 4 4 6 = 18 bits

SELA	SELB	SELD	OPR

Section D

Q12. (a) i) 400+402= 802
ii) 400+ 300= 700
iii) 401
iv) 100
v) 100
vi) 99

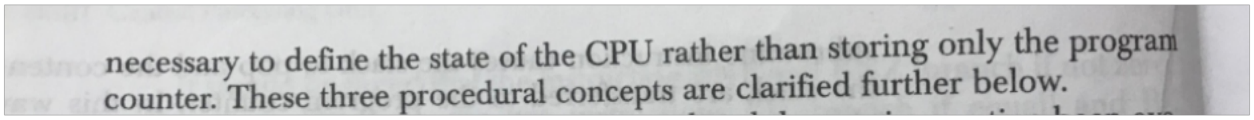
(b)



Program Interrupt

The concept of program interrupt is used to handle a variety of problems that arise out of normal program sequence. [Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request. Control returns to the original program after the service program is executed.]

The interrupt procedure is, in principle, quite similar, to a subroutine call except for three variations: (1) The interrupt is usually initiated by an internal or external signal rather than from the execution of an instruction (except for software interrupt as explained later); (2) the address of the interrupt service program is determined by the hardware rather than from the address field of an instruction; and (3) an interrupt procedure usually stores all the information



necessary to define the state of the CPU rather than storing only the program counter. These three procedural concepts are clarified further below.

Q13. (a)

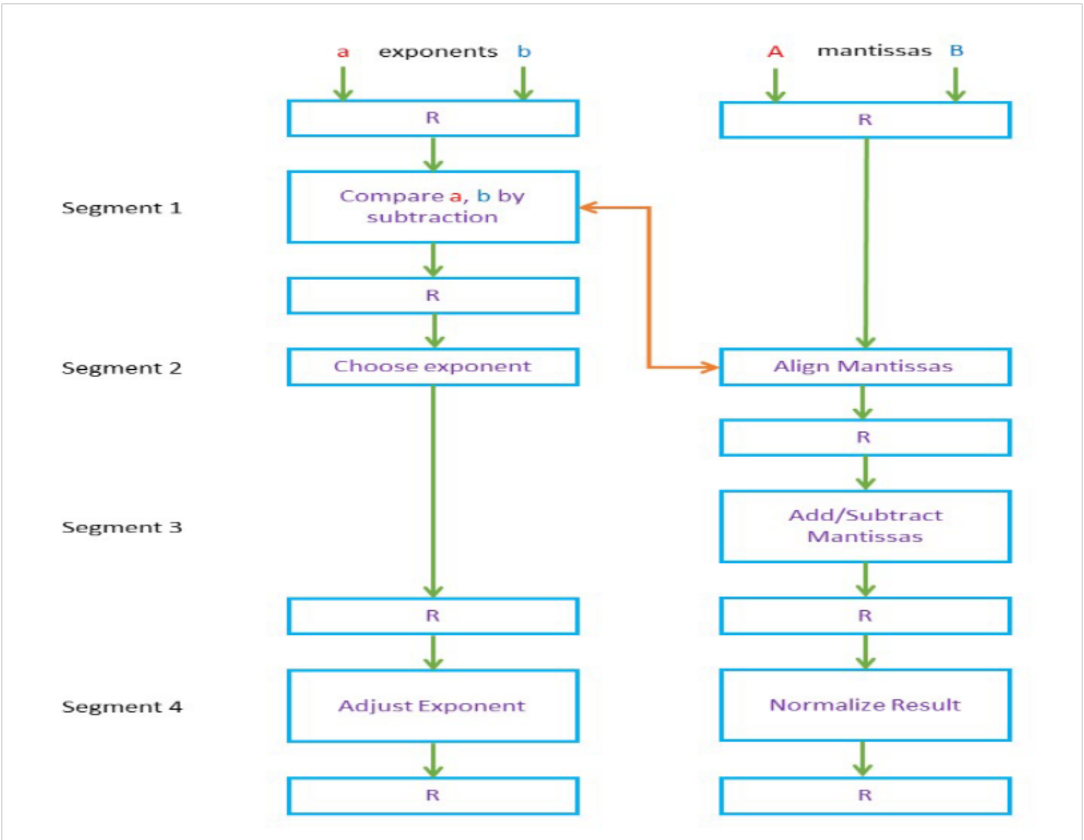


Figure | Arithmetic pipeline for floating point add/subtract operation

Example Solved:

1.) Compare the exponents

$$3-2 = 1$$

2.) Align the Mantissa

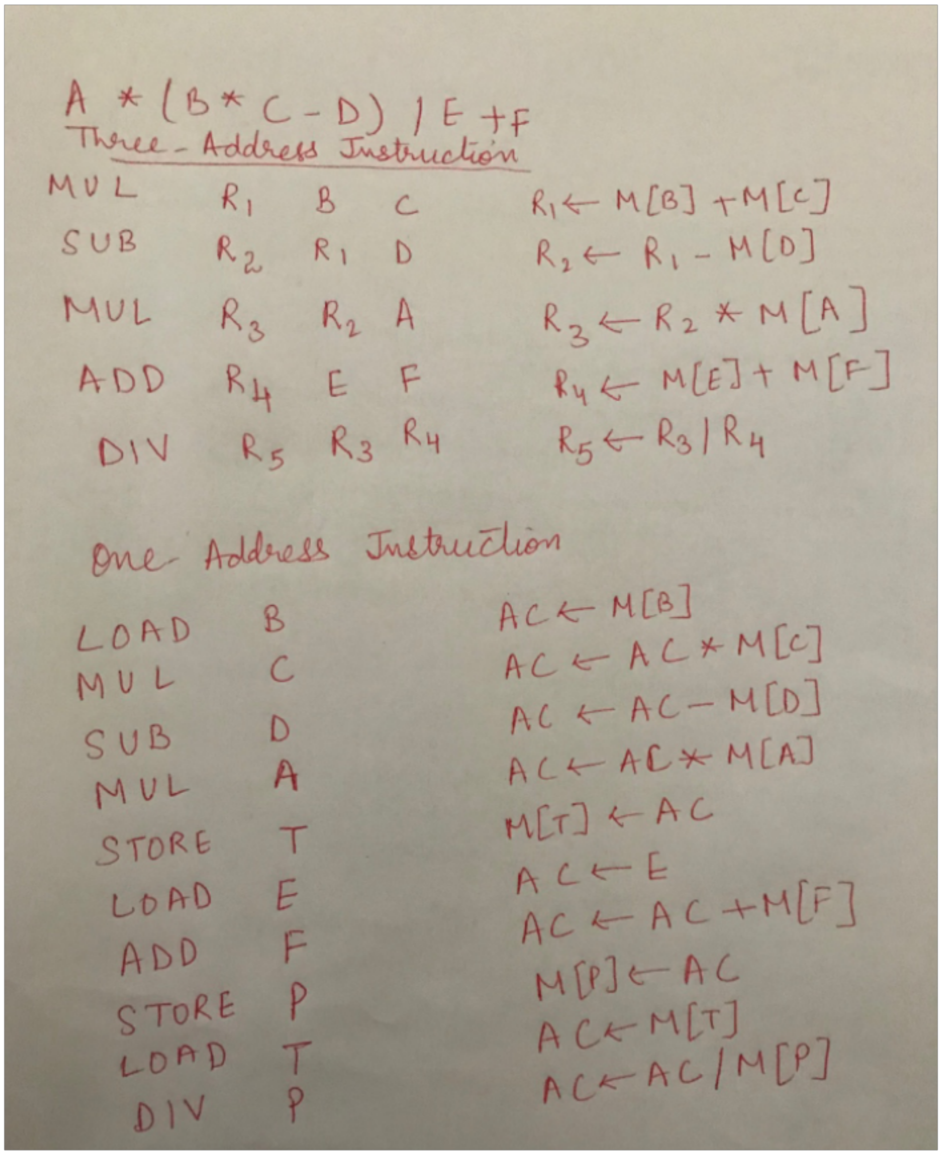
$$0.0450 * 10^3$$

3.) Add the numbers

$$0.9756 * 10^3 + 0.0450 * 10^3 = 1.0206 * 10^3$$

4.) Normalize the result Ans = $0.10206 * 10^4$

(b)



$A * (B * C - D) / E + F$

Three - Address Instruction

MUL	R_1	B	C	$R_1 \leftarrow M[B] + M[C]$
SUB	R_2	R_1	D	$R_2 \leftarrow R_1 - M[D]$
MUL	R_3	R_2	A	$R_3 \leftarrow R_2 * M[A]$
ADD	R_4	E	F	$R_4 \leftarrow M[E] + M[F]$
DIV	R_5	R_3	R_4	$R_5 \leftarrow R_3 / R_4$

One - Address Instruction

LOAD	B	$AC \leftarrow M[B]$
MUL	C	$AC \leftarrow AC * M[C]$
SUB	D	$AC \leftarrow AC - M[D]$
MUL	A	$AC \leftarrow AC * M[A]$
STORE	T	$M[T] \leftarrow AC$
LOAD	E	$AC \leftarrow E$
ADD	F	$AC \leftarrow AC + M[F]$
STORE	P	$M[P] \leftarrow AC$
LOAD	T	$AC \leftarrow M[T]$
DIV	P	$AC \leftarrow AC / M[P]$