SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
Coimbatore – 641 035.

B.E / B.Tech – Internal Assessment Exam- II
Academic Year 2023-2024 (ODD)
FIFTH SEMESTER (REGULATION R2019)
19ITT202 – COMPUTER ORGANIZATION AND ARCHITECTURE

A

TIME: 1.5 HOURS **MAXIMUM MARKS: 50**

ANSWER ALL QUESTIONS

PART A — (5 x 2 = 10 Marks)			
1.	What are the rules to perform addition on floating point numbers? 1. Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents. 2. Set the exponent of the result equal to the larger exponent. 3. Perform addition/subtraction on the mantissas and determine the sign of the result. 4. Normalize the resulting value, if necessary.	CO2	UND
2.	Subtract $(11010)_2$-$(10000)_2$ using 1's complement and 2's complement method. 1s complement-> 01001 2s complement -> 01010	CO2	APP
3.	Mention the various phase of an executing an instruction. fetching an instruction from memory, decoding the fetched instruction, reading the address from memory, and finally, instruction execution	CO3	REM
4.	What are steps required to execute an instruction by the processor? An instruction is executed by performing one or more of the following operations: 1) Transfer a word of data from one register to another or to the ALU. 2) Perform arithmetic or a logic operation and store the result in a register. 3) Fetch the contents of a given memory-location and load them into a register. 4) Store a word of data from a register into a given memory-location.	CO3	UND
5.	Write the sequence of operations to perform the Instruction [R3]<-[R1]+[R2] 1. R1out, Yin 2. R2out, SelectY, Add, Zin 3. Zout, R3in	CO3	UND

PART- B (2 x 13 = 26 Marks , 1*14=14 Marks)

6.	(a)	Perform the arithmetic operations below with binary numbers and with negative numbers in signed 2's complement representation. Use seven bits to accommodate each number together with its sign. In each case, determine if there is an overflow by checking the carries into and out of the sign bit position. a. (+35) + (+40)	13	CO2	APP

b. (-35) + (-40)
c. (-35) - (+40)

When two numbers of n digits each are added and the sum occupies n + 1 digits, we say that an overflow occurred.
A result that contains n + 1 bits cannot be accommodated in a register with a standard length of n bits.
The detection of an overflow after the addition of two binary numbers depends on whether the numbers are considered to be signed or unsigned. When two unsigned numbers are added, an overflow is detected from the end carry out of the most significant position.
In the case of signed numbers, the leftmost bit always represents the sign, and negative numbers are in 2' s complement form. When two signed numbers are added, the sign bit is treated as part of the number and the end carry does not indicate an overflow.
An overflow cannot occur after an addition if one number is positive and the other is negative, since adding a positive number to a negative number produces a result that is smaller than the larger of the two original numbers. An overflow may occur if the two numbers added are both positive or both negative.
An overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position. If these two carries are not equal, an overflow condition is produced. If the two carries are applied to an exclusive-OR gate, an overflow will be detected when the output of the gate is equal to 1 .

a) +35 0 100011
+40 0 101000
+75 1 001011

Last two carries are F=0 and E=1. F XOR E= 1.Hence overflow had occurred.

b) -35 1 011101
-40 1 011000
-75 1 110101

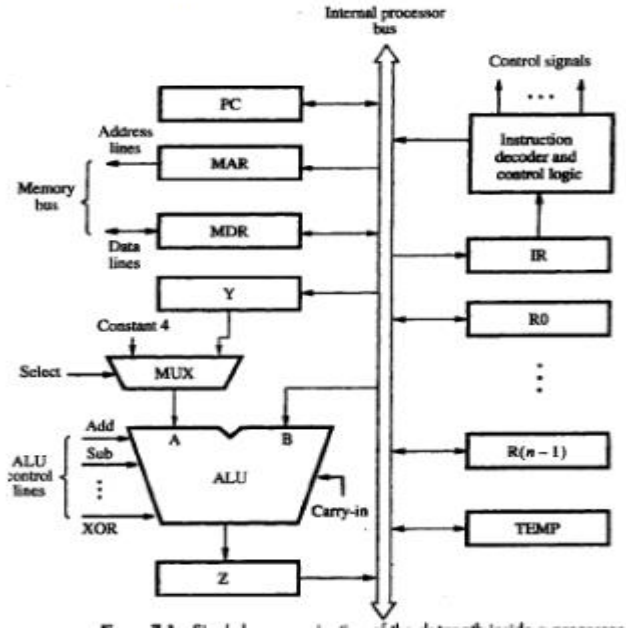
Last two carries are F=1 and E=0. F XOR E= 1.Hence overflow had occurred.

c) –(35)-(+40) . Add 2's complement of 40 to 2' complement of 35. Same as above

(OR)

(b) Write a short note on single bus organization

ALU and all the registers are interconnected via a Single Common Bus. Data & address lines of the external memory-bus is connected to the internal processor-bus via MDR and MAR respectively. (MDRà Memory Data Register, MAR à Memory Address Register). MDR has 2 inputs and 2 outputs. Data may be loaded into MDR either from memory-bus (external) or from processor-bus (internal)



MAR"s input is connected to internal-bus; MAR"s output is connected to externalbus. Instruction Decoder & Control Unit is responsible for issuing the

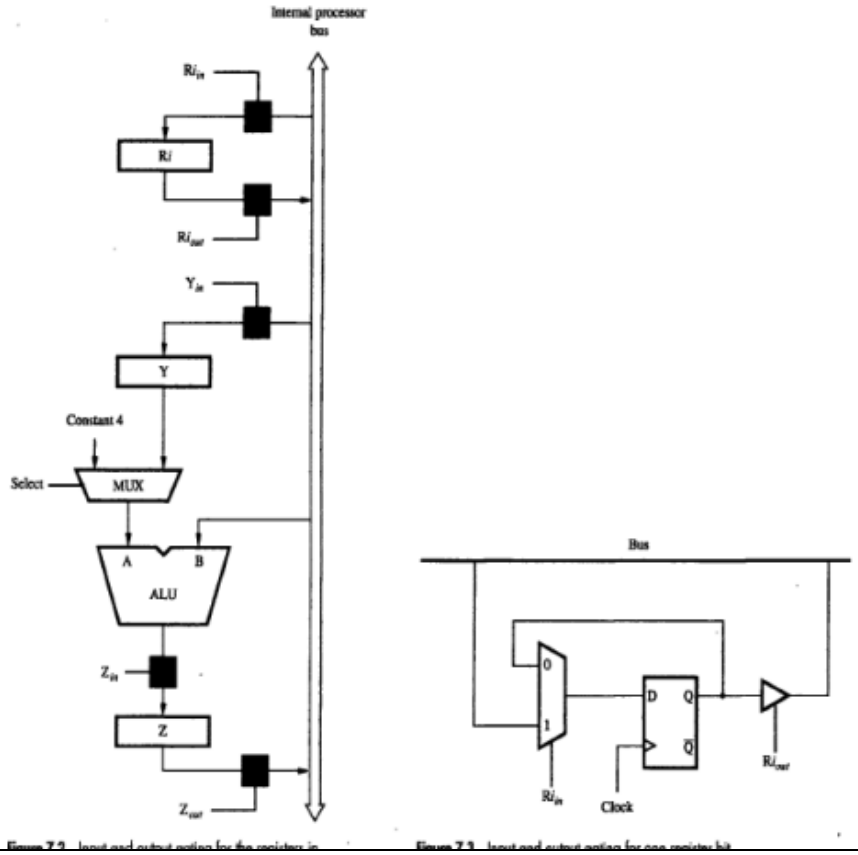
13 CO3 UND

control-signals to all the units inside the processor. We implement the actions specified by the instruction (loaded in the IR). Register R0 through R(n-1) are the Processor Registers. The programmer can access these registers for general-purpose use. Only processor can access 3 registers Y, Z & Temp for temporary storage during program-execution. The programmer cannot access these 3 registers. In ALU,1) "A" input gets the operand from the output of the multiplexer (MUX). 2) "B" input gets the operand directly from the processor-bus. There are 2 options provided for "A" input of the ALU. MUX is used to select one of the 2 inputs. MUX selects either output of Y or constant-value 4(which is used to increment PC content).

Disadvantage: Only one data-word can be transferred over the bus in a clock cycle. Solution: Provide multiple internal-paths. Multiple paths allow several data-transfers to take place in parallel.

REGISTER TRANSFERS

Instruction execution involves a sequence of steps in which data are transferred from one register to another. For each register, two control-signals are used: Riin & Riout. These are called Gating Signals. Riin=1 = data on bus is loaded into Ri. Riout=1 as content of Ri is placed on bus. Riout=0, makes bus can be used for transferring data from other registers.For example, Move R1, R2; This transfers the contents of register R1 to register R2. This can be accomplished as follows: 1) Enable the output of registers R1 by setting R1out to 1 (Figure 7.2). This places the contents of R1 on processor-bus. 2) Enable the input of register R2 by setting R2out to 1. This loads data from processor-bus into register R4.



7.	(a)	**i**) Discuss about hardwired control Hardwired control is a method of control unit design (Figure). The control-signals are generated by using logic circuits such as gates, flip-flops, decoders etc. Decoder / Encoder Block is a combinational-circuit that generates required control-outputs depending on state of all its inputs. Instruction decoder decodes the instruction loaded in the IR. If IR is an 8 bit register, then instruction decoder generates 28(256 lines); one for each instruction. It consists of a separate output-lines INS1 through INSm for each machine instruction. According to code in the IR, one of the output-lines INS1 through INSm is set to 1, and all other lines are set to 0. Step-Decoder provides a separate signal line for each step in the control sequence. Encoder gets the input from instruction decoder, step decoder, external inputs and condition codes. It uses all these inputs to generate individual control-signals: Yin, PCout, Add, End and so on. For example (Figure 7.12), Zin=T1+T6.ADD+T4.BR; This signal is asserted during time-slot T1 for all instructions during T6 for an Add instruction. During T4 for unconditional branch instruction, when RUN=1, counter is incremented by 1 at the end of every clock cycle. When RUN=0, counter stops counting. After execution of each instruction, end signal is generated. End signal resets step counter. Sequence of operations carried out by this machine is determined by wiring of logic circuits, hence the name "hardwired". Advantage: Can operate at high speed. Disadvantages: 1) Since no. of instructions/control-lines is often in hundreds, the complexity of control unit is very high. 2) It is costly and difficult to design. 3) The control unit is inflexible because it is difficult to change the design.	7 6	CO3	UND

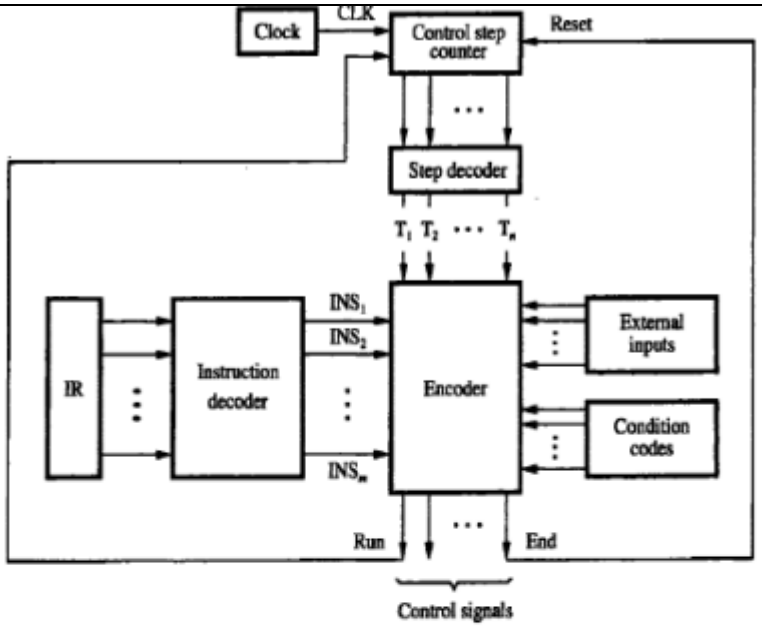
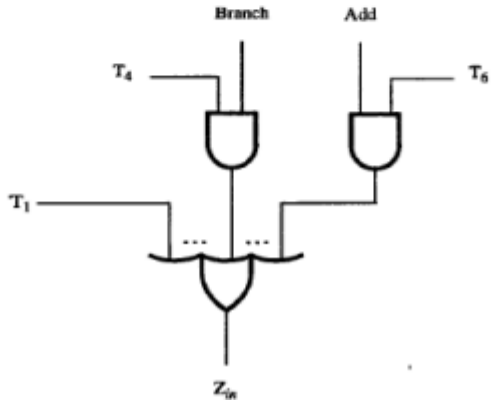
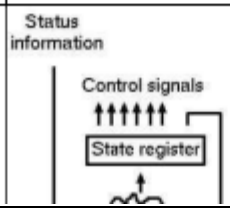
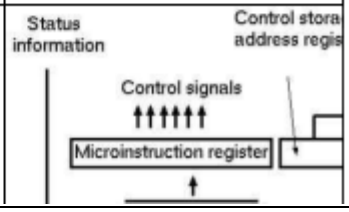


Figure 7.11 Separation of the decoding and encoding functions.



ii) Differentiate hardwired control and microprogrammed control

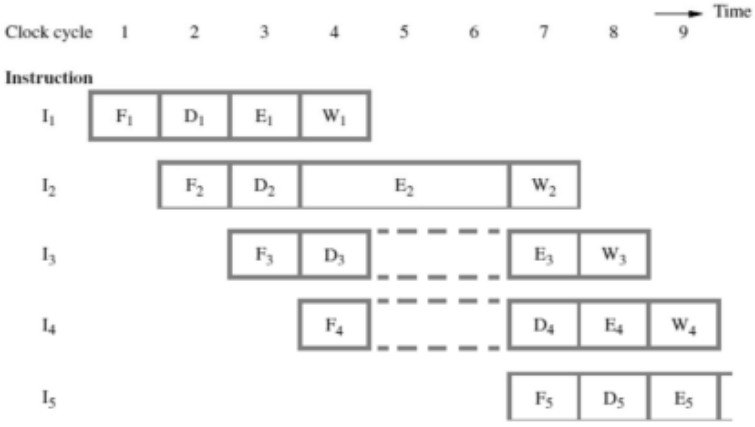
HARDWIRED CONTROL VS MICROPROGRAMMED CONTROL

Attribute	Hardwired Control	Microprogrammed Control
Definition	Hardwired control is a control mechanism to generate control-signals by using gates, flip- flops, decoders, and other digital circuits.	Micro programmed control is a control mechanism to generate control-signals by using a memory called control store (CS), which contains the control-signals.
Speed	Fast	Slow
Control functions	Implemented in hardware.	Implemented in software.
Flexibility	Not flexible to accommodate new system specifications or new instructions.	More flexible, to accommodate new system specification or new instructions redesign is required.
Ability to handle large or complex instruction sets	Difficult.	Easier.
Ability to support Operating systems& diagnostic features	Very difficult.	Easy.
Design process	Complicated.	Orderly and systematic.
Applications	Mostly RISC microprocessors.	Mainframes, some microprocessors.
Instruction set size	Usually under 100 instructions.	Usually over 100 instructions.
ROM size	-	2K to 10K by 20-400 bit microinstructions.
Chip area efficiency	Uses least area.	Uses more area.
Diagram		

(OR)

(b) Explain about data hazards with an exampl**e.**

A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result some operation has to be delayed, and the pipeline stalls



Effect of an execution operation taking more than one clock cycle

Consider a program that contains two instructions, I1 followed by I2. When this program is executed in a pipeline, the execution of I2 can begin before the execution of I1 is completed. This means that the results generated by I1 may not be available for use by I2. We must ensure that the results obtained when instructions are executed in a pipelined processor are identical to those obtained when the same

1 3 | CO3 | REM

instructions are executed sequentially. The potential for obtaining incorrect results when operations are performed concurrently can be demonstrated by a simple example. Assume that A=5, and consider the following two operations:

$A \leftarrow 3 + A$

$B \leftarrow 4 \times A$

When these operations are performed in the order given, the result is B = 32. But if they are performed concurrently, the value of A used in computing B would be the original value, 5, leading to an incorrect result. If these two operations are performed by instructions in a program, then the instructions must be executed one after the other, because the data used in the second instruction depend on the result of the first instruction. On the other hand, the two operations

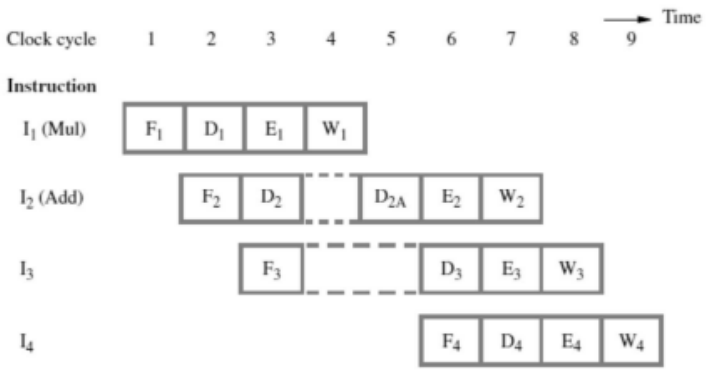
$A \leftarrow 5 \times C$

$B \leftarrow 20 + C$ can be performed concurrently, because these operations are independent.

This example illustrates a basic constraint that must be enforced to guarantee correct results. When two operations depend on each other, they must be performed sequentially in the correct order. This rather obvious condition has far-reaching consequences. Understanding its implications is the key to understanding the variety of design alternatives and trade-offs encountered in pipelined computers. Consider the pipeline in Figure 2. The data dependency just described arises when the destination of one instruction is used as a source in the next instruction.

For example, the two instructions Mul R2,R3,R4 Add R5,R4,R6 give rise to a data dependency. The result of the multiply instruction is placed into register R4, which in turn is one of the two source operands of the Add instruction. Assuming that the multiply operation takes one clock cycle to complete, execution would proceed as shown in Figure.

As the Decode unit decodes the Add instruction in cycle 3, it realizes that R4 is used as a source operand. Hence, the D step of that instruction cannot be completed until the W step of the multiply instruction has been completed. Completion of step D2 must be delayed to clock cycle 5, and is shown as step D2A in the figure. Instruction I3 is fetched in cycle 3, but its decoding must be delayed because step D3 cannot precede D2. Hence, pipelined execution is stalled for two cycles.

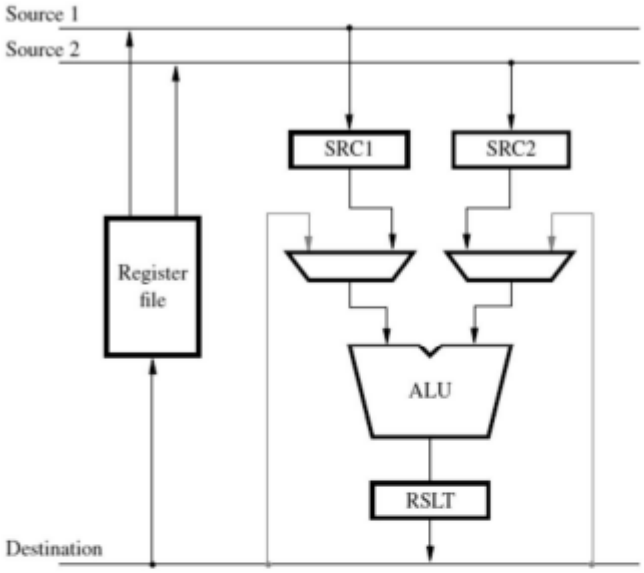


Operand Forwarding

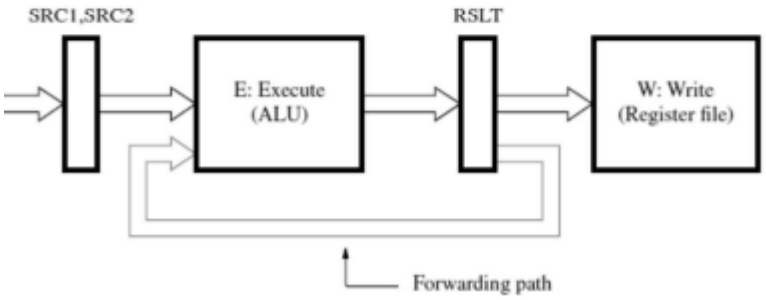
The data hazard just described arises because one instruction, instruction I2 in Figure, is waiting for data to be written in the register file. However, these data are available at the output of the ALU once the Execute stage completes step E1.

Hence, the delay can be reduced, or possibly eliminated, if we arrange for the result of instruction I1 to be forwarded directly for use in step E2. Figure a shows a part of the processor datapath involving the ALU and the register file. This arrangement is similar to the three-bus structure in Figure, except that registers SRC1, SRC2, and RSLT have been added. These registers constitute the interstage buffers needed for pipelined operation, as illustrated in Figure b. With reference to Figure b, registers SRC1 and SRC2 are part of buffer B2 and RSLT is part of B3. The data forwarding mechanism is provided by the blue connection lines. The two multiplexers connected at the inputs to the ALU allow the data on the destination bus to be selected instead of the contents of either the SRC1 or SRC2 register.

When the instructions in Figure are executed in the datapath of Figure, the operations performed in each clock cycle are as follows. After decoding instruction I2 and detecting the data dependency, a decision is made to use data forwarding. The operand not involved in the dependency, register R2, is read and loaded in register SRC1 in clock cycle 3. In the next clock cycle, the product produced by instruction I1 is available in register RSLT, and because of the forwarding connection, it can be used in step E2. Hence, execution of I2 proceeds without interruption.

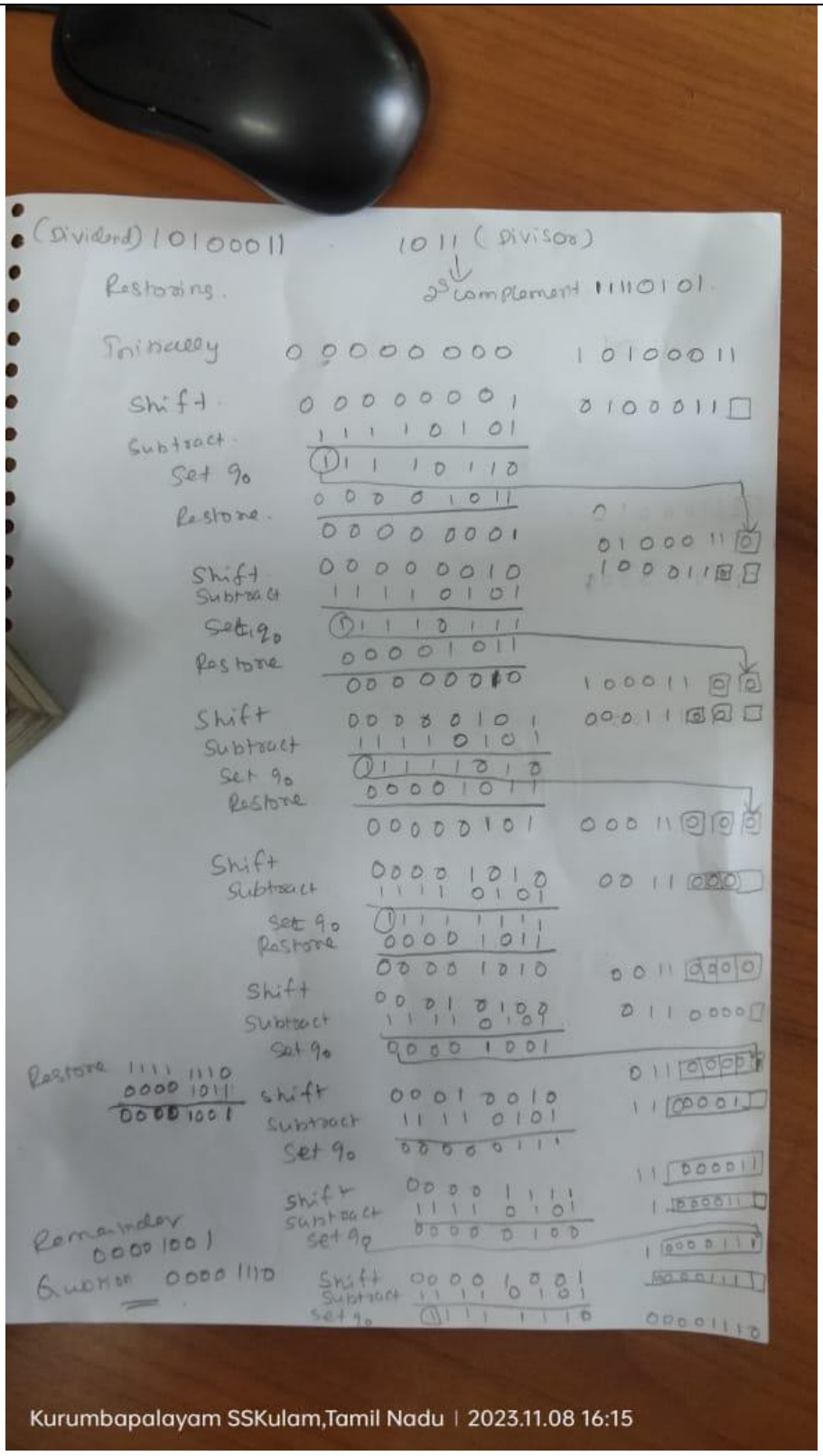


(a) Datapath



(b) Position of the source and result registers in the processor pipeline

8.	(a)	Show the step-by-step multiplication process using Booth algorithm when the following binary numbers are multiplied. Assume 5-bit registers that hold signed numbers. The multiplicand in both cases is + 15. a. (+ 15) * (+ 13) b. (+ 15) * (- 13) a. 00110 00011 +195 b. 11001 11101 −195	14	CO2	APP
		(OR)			
	(b)	Divide using the restoring and non-restoring division algorithm with step by step intermediate results and explain. (a) 10100011 by 1011 (b) 00001111 by 0011. (Use a dividend of eight bits.)	14	CO2	APP


(Dividend) 10100011
1011 (Divisor)
Restoring.
2's complement 11110101
Initially
Shift
Subtract
Set q0
Restore
Shift
Subtract
Set q0
Restore
Shift
Subtract
Set q0
Restore
Shift
Subtract
Set q0
Restore
Shift
Subtract
Set q0
Restore 1111 1110
0000 1011
0000 1001
Shift
Subtract
Set q0
Shift
Subtract
Set q0
Remainder 0000 1001
Quotient 0000 1110
Shift
Subtract
Set q0
Kurumbapalayam SSKulam,Tamil Nadu | 2023.11.08 16:15