Computer Architecture Basic

Computer architecture is a set of rules and methods that describe the functionality, organization, and implementation of computer systems.

Types of Computer Architecture (categories Sub)

- **Instruction Set Architecture or Isa.**: whenever an instruction is given to processor, its role is to read and act accordingly.

- **Micro Architecture**: It describes how a particular processor will handle and implement instructions for Isa. (Isa instruction handle src)

- **System design**: It includes the other entire hardware components within the System such as multiprocessing. (other component src hardware)

#Importance of Computer Architecture.

• The main role of CA is to balance the performance, efficiency, cost and reliability of a Computer system. For Example, It acts as a bridge between Computer software and hardware It works as a programer's view as a machine. It is communicate users and Computer.

• Diffirence Between Computer Architecture and organization

Computer Architecture	Computer Organization.
• It describes what the computer does.	• It describes how it does it.
• Deals with the functional behavior of computer system	• Deals with a structural relationship.
• Deals with high lvl design issues	• Deals with low level design issues.
• As a programmer you can view architecture as a series of instruction addressing modes, registers.	• The implementation of the architecture is called organization.

Computer Performance

Computer performance is the amount of work accomplished by a computer system. It is basically depends on
- response time
- throughput
- Execution time of a computer system.

response time: Time between start and completion of a task. also called wall clock time.

Response time = Cpu time + waiting time (I/O, scheduling)

Cpu Execution time: Time spent executing the program instructions.

Cpu time = User Cpu time + Kernel Cpu time

- Can be related to the number of cpu clock cycles.

Throughput: Total work done per unit of time.

Being use of faster version of a processor Decreasing the execution time it improves. throughput:

Relative Performance: It is the ratio between two computer (Suppose A and B) Performance a

$$\text{Relative performance} = \frac{\text{Performance A}}{\text{Performance B}} \text{ on.}$$

$$\Downarrow$$

$$\frac{\text{execution time B}}{\text{execution time A}} = n$$

Clock cycle: Cpu speed is determined the clock cycle. which is the amount of t between two pulses on an oscillator.

Clock rate: $\text{Clock rate} = \frac{1}{\text{Clock cycle}}$.

measurement of Clock cycles per second

ENIAC: Electric Numerical Integrator and Computer. It is one of the earliest electronic general purpose computers. It was Turing-Complete and able to solve a large class of numerical problems. It is build during world war II by the US.

Abacus: An abacus is a mechanical device that is used to calculate the arithmetic calculations. Quickly. It is also referred to as a counting frame. Today is widely used in brain development programs. The standard abacus can be used to perform addition, subtraction, multiplication and division. and also be used to extract square roots and cubic-roots

reduced Instruction set computer

complex instruction set computer

Difference Between RISC and CISC.

RISC	CISC
• Emphasis on Software	• Emphasis on hardware
• Small number of fixed length instructions.	• Large number of instructions.
• Single clock cycle instructions.	• Instructions can take several clock cycles.
• Heavy use of Ram memory registers	• More efficient use of Ram.
• Execution time is very Short	• Execution time is very longer.

• Thus it can be said RISC is better as it also can be designed quickly than CISC.

Scale of ICs

total transistors.

SSI - Small scale integration → Less than 100

MSI - Medium Scale " → Between 100 and 1000

LSI - Large " " → " 1000 and 10,000

VLSI - Very " " " → " 10,000 and 100,000

ULSI - ultra Large " " → " 100,000 and 10,000,000

better

Performance improved করার = reducing clock

Clock cycles per instructions (CPI): Elaborated as clock cycles per instruction

Average CPI for a given program = $\frac{\text{total CPU clock cycles}}{\text{Total instruction count}}$

MIPS: Millions Instructions Per Second (execution rate)

* Faster machine = Larger MIPS.

$$MIPS = \frac{\text{instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{clock rate}}{CPI \times 10^6}$$

Drawbacks of MIPS:

- Does not take into account the capability of instructions. (instruction count different)
- MIPS varies between programs on the same computer. (single program এর ক্ষেত্রে)
- A higher MIPS rating doesn't mean better performance.

Cpu Clock cycles = Introduction for a program × Average

CPU time अथवा

CPU Time = $\frac{\text{Cpu clock cycles}}{\text{Clock rate}}$

CPU Execution Time = Cpu clock cycles × Clock cycle time

= Introduction for program × CPI × clock cycle time

clock cycle time = $\frac{1}{\text{Clock rate}}$

$$= \frac{\text{Introduction count (for program)} \times \text{CPI}}{\text{Clock rate}}$$

Clock rate = $\frac{\text{Introduction for program} \times \text{CPI}}{\text{Cpu Execution Time}}$

Instruction Per Second (IPS) = $\frac{\text{Clock Rate}}{\text{CPI}}$

Clock cycle = execution time × clock rate

CPI = $\frac{\text{Clock cycles}}{\text{Instruction Count}}$ = cpu c.cy = instruction × CPI

Overall CPI = $\frac{\sum_i CPI_i \times L_i}{\text{instruction count}}$

MIPS = $\frac{\text{Clock rate}}{CPI \times 10^6}$

Cpu clock cycle = $\sum$ (CPI × count)

Ex: Computer A has a Clock cycle time 250ps and 2.0 for some prog

" B " " " " 500ps and 1.2 for some prog.

which is faster and why?

Ans we know that

CPU clock cycle = no. of instruction program × CPI

So A have = I × 2.0

B " = I × 1.2

CPU execution time = CPU clock cycles × clock times

for A = I · 2.0 × 250ps = 500Ips

" B = I · 1.2 × 500 ps = 600Ips

So A is faster becz 500Ips low time needed.

amount of faster

$$\frac{\text{CPU performance A}}{\text{CPU performance B}} = \frac{\text{Execution time B}}{\text{Execution time A}} = \frac{600\text{Ips}}{500\text{Ips}}$$

$$= 1.2.$$

Example 2

Computer
A have a clock cycle time 2 ns CPI 3.0 for prgm
B " " " " 3ns " 2.0 " "

Ans

here
Cpu Clock cycles = Instruction for program × CPI

A = I × 3.0

B = I × 2.0

Execution time = cpu clock cycles × cpu clock time

A = I · 3.0 × 2ns = 6Ins

B = I · 2.0 × 3 = 6ns.

So both showing same performance level.

Example 3

Consider there different processors P1, P2, P3 executing the same instruction set.

P1 has a 3 GHz clock rate and CPI 1.5

P2 " " 2.5 GHz clock rate and CPI 1.0

P3 " " 4.0 GHz " " " CPI of 2.2

1. which one has the highest performance expressed in instructions per second.

2. If the processor each execute a program in 10s, find the number of cycles and the number of instruction.

3. trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. what clock rate should we have to get this time reduction.

Ans ① $IPS = \frac{\text{Clock Rate}}{CPI}$

for P_1 :

$$IPS = \frac{\text{Clock Rate}}{CPI} = \frac{3\ GHz}{1.5} = \frac{3 \times 10^9}{1.5}$$

$= 2 \times 10^9$ instruction per second. (i PS).

for $P_2 = \frac{2.5 \times 10^9}{1.0} = 2.5 \times 10^9$ iPS

$P_3 = \frac{9 \times 10^9}{2.2} = 1.82 \times 10^9$ iPS

Here P2 = 2.5×10^9 iPS is biggest performance.

② Number of cycles:

Cpu execution times = Cpu Clock cycles × clock time

$$\text{Cpu clock cycles} = \frac{\text{Cpu execution time}}{\text{Clock cycle time}}$$

= Cpu execution time × $\frac{1}{\text{Clock cycle time}}$

= Cpu execution time × Clock rate.

So

Clock cycles = execution time × clock rate

for A " = $10 \times 3 \times 10^9 = 3 \times 10^{10}$ cycles

" B " = $10 \times 2.5 \times 10^9 = 2.5 \times 10^{10}$ cycles

" C " = $10 \times 4 \times 10^9 = 4 \times 10^{10}$ cycles

Number of instructions: CPI = $\frac{\text{CPU clock cycles}}{\text{Instruction count}}$

So

Instruction count = $\frac{\text{Clock cycles}}{\text{CPI}}$

for P1 P0 = $\frac{3 \times 10^{10}}{1.5} = 2 \times 10^{10}$

P2 " = $\frac{2.5 \times 10^{10}}{1.0} = 2.5 \times 10^{10}$

P3 " = $\frac{4 \times 10^{10}}{2.2} = 1.82 \times 10^{10}$

Q3

Ans (3)

$$\text{Execution time} = \frac{\text{num of instruction} * \text{CPI}}{\text{clock rate}}$$

we want reduce execution time 30% and CPI increase 20% then

$$\frac{30}{100-30} = 70\% = 0.7$$

$$\frac{(100+20)}{} = 120\% = 1.2$$

$$\text{Execution time} * 0.7 = \frac{\text{instruction} * \text{CPI} * 1.2}{\text{new clock rate}}$$

$$\text{So new clock rate} \Rightarrow \frac{\text{instruction} * \text{CPI} * 1.2}{\text{Execution time} * 0.7}$$

$$\text{So} = \boxed{\text{clock rate} \times \frac{1.2}{0.7}}$$

$P_1 = 3 * 1.71 = 5.13$ GHz

$P_2 = 2.5 * 1.71 = 4.27$ GHz

$P_3 = 4\ GH_2 * 1.71 = 6.84$ GHz

Example 4

Class	ALU	FPU	Load	Store	Branch
frequency	20%	30%	20%	10%	20%
CPI	1	4	5	3	2

a) what is average CPI

Ans Average CPI : $0.2*1 + 0.3*4 + 0.2*5 + 0.1*3 + 0.2*2 = 3.1$

Percent time, ALU এর জন্য কত

$$ALU = \frac{0.2*1}{3.1} = 6.4\%$$

$$FPU = 1.2/3.1 = 38.7\%$$

$$Load = (0.2*5)/3.1 = 32.3\%$$

$$Store = (0.1*3)/3.1 = 9.7\%$$

$$Branch = (0.2*2)/3.1 = 12.9\%$$

MIPS Math

Measurement	A	B
Instruction count for program	→ 10B	→ 8B
Clock rate	→ 4.2 GHz	→ 4.0 GHz
CPI	→ 1.0	→ 1.1

Q: which is higher rating? faster?

- we know MIPS = $\frac{\text{Clock rate}}{\text{CPI} \times 10^6}$

for A = $\frac{4.2 \times 10^9}{1.0 \times 10^6}$ = 4200 MIPS

" B = $\frac{4.0 \times 10^9}{1.1 \times 10^6}$ = 3636 MIPS

CPU time (or) Execution time = $\frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}}$

for A = $\frac{10 \times 10^9 \times 1.0}{4.2 \times 10^9}$ = 2.38

" B = $\frac{8 \times 10^9 \times 1.1}{4.0 \times 10^9}$ = 2.20 sec

- Computer A is higher MIPS, B is low execution time

Instruction

The word of machine language are instructions.

High programmer $A = B + C$ (in computer binary form)

operation of computer hardware:

$$f = (g+h) - (i+j)$$

add to g h } to compute g+h and

add t1 i j } t1 = i, j

sub f t0, t1 } sub to - t1

formats.

There are 3 instructions

I format → | OP | rs | rt | rd | shamt | funct | ← 32

J " → | OP | rs | rt | 16 bit address | ← 32 (16+16=32)

R " | OP | 26 bit address | ← 32 (=26+6 or 32)

MIPS design Principle

#Simplicity favors regularity.

- keeping all instruction single size
- requiring 3 register operands arithmetic instruction

Example HLL $a = b + c$, $d = a - e$ } m/c to MIPS machine performed by compiler

add a, (b, c)

operand

destination statement both are

#Smaller is faster

$$f = (g + h) - (i + j)$$

register for $

So f, g, h, i, j assigned register $s0, $s1, $s2, $

$s4 respectfully.

$t0, $t1 temporary variable.

(Size of register MIPS = 32 bit)

add $t0, $s1, $s2 [g+h in t0]

add $t1, $s3, $s4 [i+j in t1]

sub $s0, $t0, $t1 [f=(g+h)-(i+j)]

good design demand good compromise

- compromise between providing for large address.
- Keeping all instructions same length

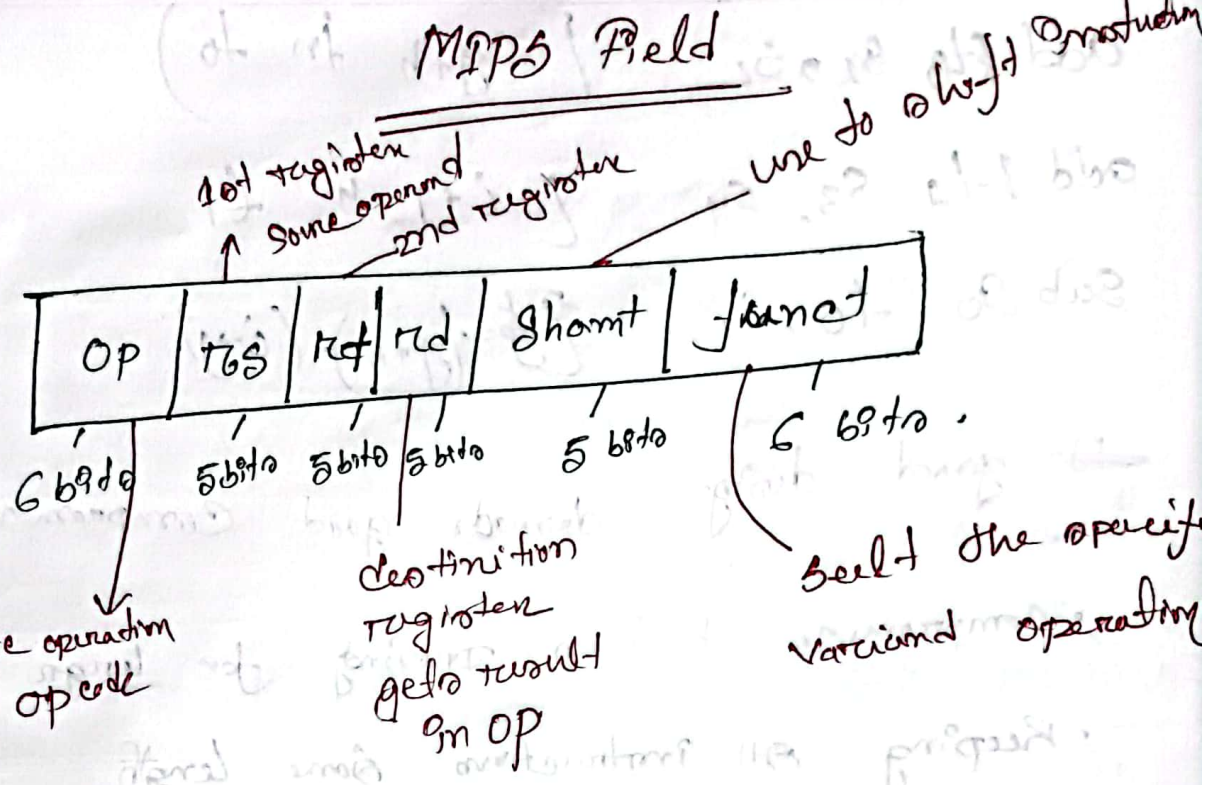
Make the common case fast

Quick add instruction with one constant operand is called add immediate or addi

to add 4 to register $s3, we write:

addi $s3, $s3, 4 (# $s3 = $s3 + 4)

MIPS Field



Example : Add $t0 $s_1, $s_2.

represent in MIPS field in binary and decimal

let $s_1 = 17$, $s_2 = 18$, $t0 = 8$.

	OP	rs	rt	rd	shamt	funct
decimal	0	17	18	8	0 (unused)	32
binary	000000 (6 bit)	10001	10010	01000	00000	10000 (6 bit)
operation	Add	s_1	s_2	$t0	unused	32

MIPS function 32 bit long

The numeric version of the instructions are Machine language or machine code.

The layout of the instruction is instruction format.

MIPS operands

$s0	16
$s1	17
$s2	18
$s3	19
$s4	20
$s5	21
$s6	22
$s7	23

$t0	8
$t1	9
$t2	10
$t3	11
$t4	12
$t5	13
$t6	14
$t7	15

32 registers >> $zero always equals 0

$reserved large constants

MIPS uses byte addressed

	register number
$zero => constant value 0	0
$v0 - $v1 => expression evaluation	=> number 2-3
$a0 - $a3 => arguments	=> reg number 4-7
$t0 - $t7 => temporaries	=> reg num 8-15
$s0 - $s7 => Saved	=> reg num 16-23
$t8 - $t9 => temporaries	" => 24-25
$gp => global pointer	=> 28
$sp => Stack pointer	=> 29
$fp => frame pointer	=> 30
$ra => return adress	=> 31
k0 - k1 => reserved os	=> 26-27
=> assembler	=> 01

register number

MIPS assembly language

Arithmatic
- add → add $s1 $s2 $s3 ⇒ $s1 = $s1 + $s2
- subtract → sub $s1 $s2 $s3 ⇒ $s1 = $s1 - $s2

Data on register

Data transfer
- Load word → lw $s1, 100($s2) ⇒ $s1 = Memory [$s2 + 100]
- Store word → sw $s1, 100($s2) ⇒ Memory [$s2 + 100] ⇒ $s1

Data from register to memory.

Conditional Branch
- branch on equal → beq $s1, $s2, L → if ($s1 == $s2) go to L
- branch on not equal → bne $s1, $s2, L → if ($s1 ≠ $s2) go to L
- Set on less than → slt $s1, $s2, $s3 → if ($s2 < $s3) $s1 = 1 else $s1 = 0

Unconditional Jump

- Jump → J 2500 → go to 70000. (target address)
- Jump register → Jr rca → go to rca
- Jump and link → Jal 2500 → rca = PC + 4 go to 10000

MIPS machine language.

Name	format	OP	rs	rt	rd	Example shamt	funct.	comments
add	(R)	0	18	19	17	0	32	add $s1 $s2 $s3
Sub	(R)	0	18	19	17	0	34	Sub $s1 $s2 $s3
Lw	I	35	18	17	100			Lw $s1 100($s2)
Sw	I	43	18	17	100			Sw $s1 100($s2)
beq	I	4	17	18	25			beq $s1, $s2, 100
bne	I	5	17	18	25			bne $s1 $s2, 100
Slt	R	0	18	19	17	0	42	Slt $s1, $s2, $s3.
J	(J)	2	2500.					J 10000
Jr	R	0	31	0	0	0	8	Jal 10000

Short note

OP মান Lw = 35, Sw = 43

beq = 4 bne = 5

add = 8.

Short note =

funct মান add = 32

Sub = 34 Slt = 42

Jr = 8.

Example $A = b + c$ ← inc

add $s0 $s1 $s2 ← assembly

add rd, rs, rt → in mips field

machine code

0	17	18	16	0	32	← decimal
000000	10001	10010	10000	00000	100000	→ binary
op	rs	rt	rd	shamt	funct	

R format instructions

- op have 0

op	rs	rt	rd	shamt	funct
bit 6	5	5	5	5	6

- add, addu, sub, subu, and, or, nor

slt, sltu, all are R format

sll $s1 $s2 10

op	rs	rt	rd	shift Amt	funct
0000	10010	10001	01000	00010	000000

rd → 8, rt → 17, shamt → 2

I format introduction

- Have 2 registers and a constant value (immidiate) present

OP (6 bit)	rs (5 bit)	rd (5 bit)	immidiate (16 bit)

addi, addiu, sw, lw, beq, bne are I format

Example: addi $t2, $s3, 4 (2 register $t2 $s3 → 10 19)

OP	rs	rd	immediate
8	19	10	4

J format introduction:

Have an address (part of one, accetuly)

J, JAL are J format

OP	address
2	25000

J 10000

in J format address find by 10000/4 = 25000 এটা বারো 4 দিয়ে ভাগ

Practise — MIPS Instruction → Machine code

(1) bne $s3, $s4 → 1000

Go to I format

OP	RS	rt	Immidiate
bne = 5	$s3 = 19	$s4 = 20	1000
000101	10011	10100	0000 0000 0000 1000

(2) add $s0, $s1, $s2

Go to R format

OP	rs	rt	rd	Shamt	funct
add = 0	$s1 = 17	$s2 = 18	$s0 = 16	0	32
000000	10001	10010	10000	00000	100000

always 0

(3) J 1101000

J to j format

OP	address
J = 2	0000, 0000 0000 0000 0001 1010 00

000010

4. Sub $s0 $s1 $s2

q to R format

op	rs	rt	rd	shamt	funct
sub 0	$s1 17	$s2 18	$s0 16	0	sub = 34
00000	10001	10010	10000	00000	100010

SP23 Q2, b

(i) add $s5 $t5 , 20 (q to R format)

op	rs	rt	rd	samt	funct	Immediate
add = 0	$t5 = 13	0	$s5 = 21	0	32	20
000000	01101	00000	10101	00000	100000	0000 0000 0001 0100 (16 bit)

(ii) bne $s3 $s1 100

q to i format

op	rs	rt	immediate
bne = 5	$s3 = 19	$s1 = 17	100
000101	10011	10001	0000 0000 0000 0100

(iii) sw $s7, 80($s2) (I-type i-format)

here Memory ($s2 + 80) = $s7.

destination reg 2nd

op	rs	rt	immidiate
sw = 43	$s2 = 18	$s7 = 23	80
101011	10010	10111	0000 0000 0101 0000

(iv) beq $s1, $s6, 24 ($s1 == $s2 : go to 24)

I-type, I format

op	rs	rt	immidiate
beq 4	$s1 17	$s6 22	24.
000100	10001	10110	0000 0000 0001 1000

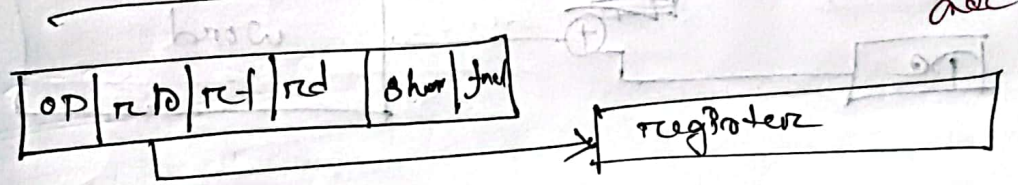
(v) sll $s1 $s2 10 (R-format)

OP	rs	rt	rd	shamt	funct
000	$0010	10001	01010	0000	
	18	17	10		
	$s2	$s1			

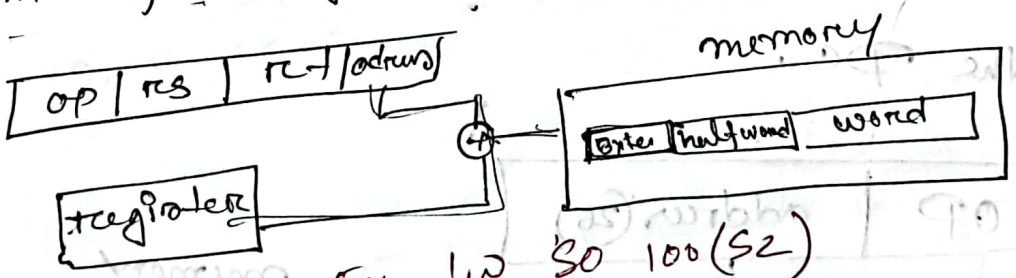
MIPS Addressing Mode

multiple forms of addressing are generically called addressing modes.

1. Register addressing ⇒ where operand is register

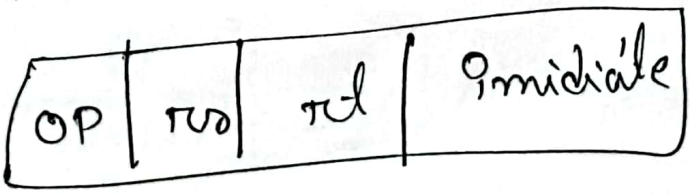


2. Based or displacement addressing ⇒ where operand is at the memory location whose address is the sum of a register and a constant in instruction.



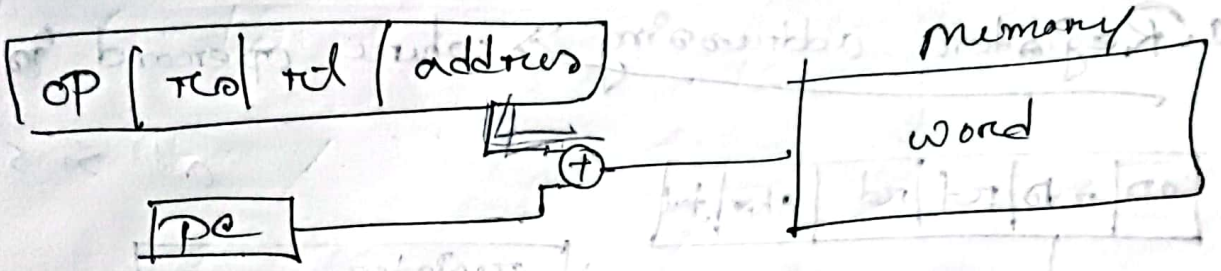
Ex lw $s0 100($s2)

3. immediate addressing ⇒ where the operand is a constant within the instruction itself.

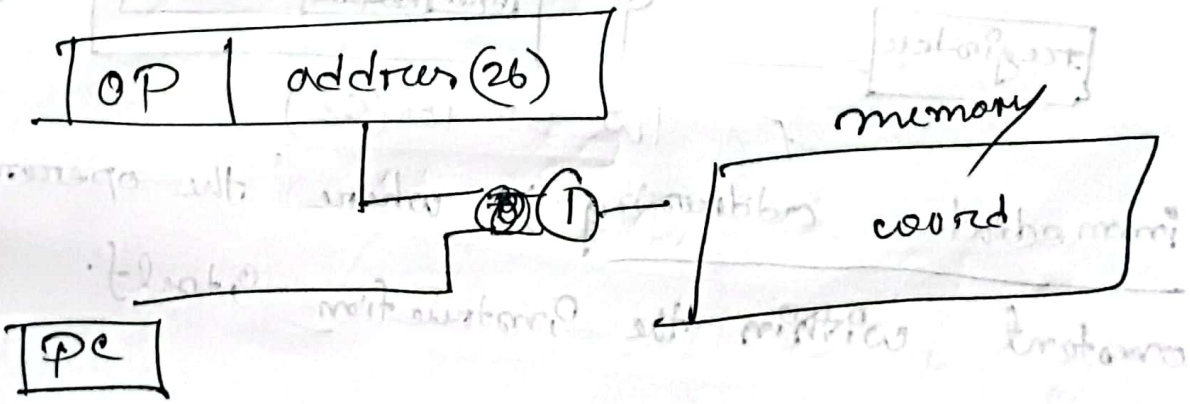


addi $s0 $s1

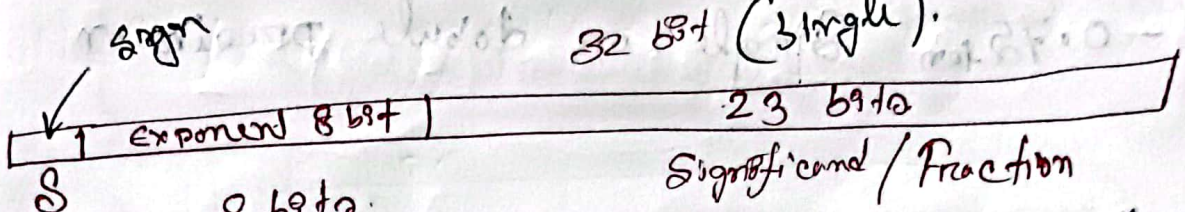
④ PC-relative addressing mode >> where the address is the sum of the PC and constant in the instruction.



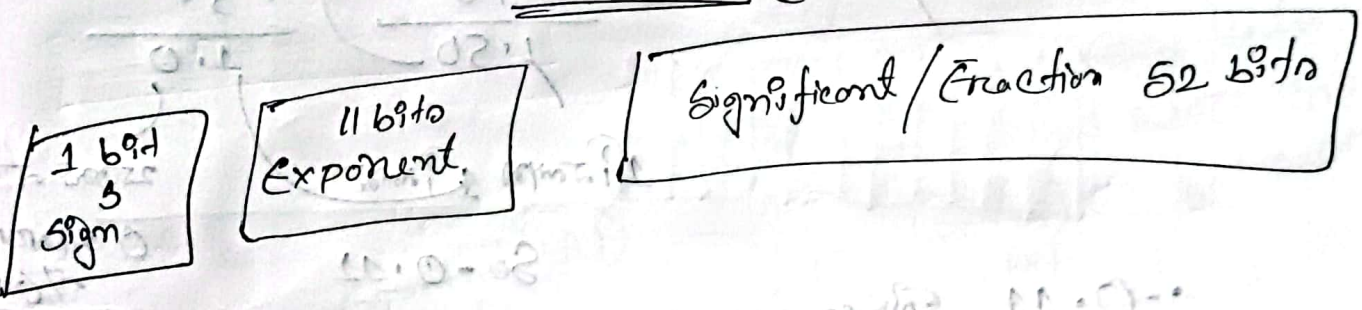
⑤ Pseudodirect addressing >> where the jump address is the 26 bits of the instruction concatenated with the upper bits of the PC



Floating Point



64 bits (double)



#Bais (0123456) so n=7

$n = 8 = 2^{n} - 1 = 2^{7} = 128 - 1 = 127$ (32 bits)

$n = 11 = 2^{n} - 1 = 2^{10} = 1024 - 1 = 1023$ (64 bits)

(0 to 11)

Example

• IEEE 754 binary representation of number -0.75_{ten} Single or double precision

-0.75

এর binary করতে

$.75 \times 2 = 1.50$

$.50 \times 2 = 1.0$

1 নিলাম 1 নিলাম

শেষে 0 আসলে শেষ করে দিব

So -0.11

• -0.11 কীভাবে আসল

চারটা 16 দশমিক আনতে

$0.11 \rightarrow 2^{-}$ (যতবার বাম)

$2^{}$ (যতবার ডান)

So here $0.11_2 \times 8 = 1.1$

$= -1.1 \times 2^{-1}$ [যতবার → তারি So 2^{-1}).

32 bit bcuz = 127

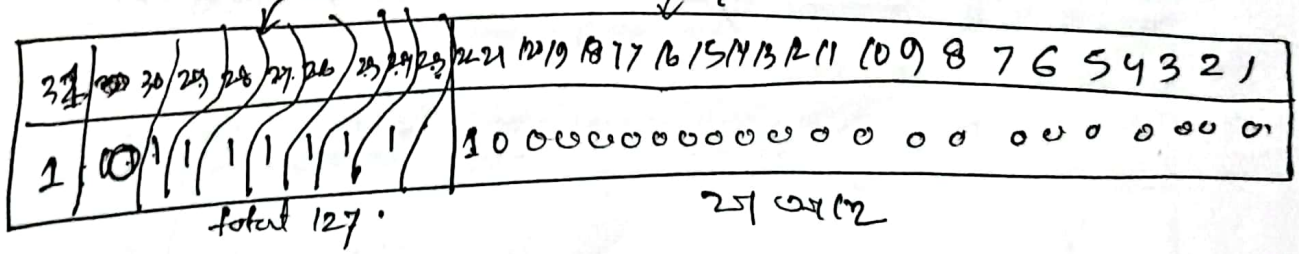
64 " " = 1023.

Single precision eqn:

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{bais})}$$

$$= (-1)^1 \times (1 + .100\ldots0) \times 2^{(126-127)}$$

$$= 1$$



On this eq exponent 126 binary করলে

128 64 32 16 8 4 2 1

0 1 1 1 1 1 1 0

64 bit (11) bit

$2^{10} = 1024$

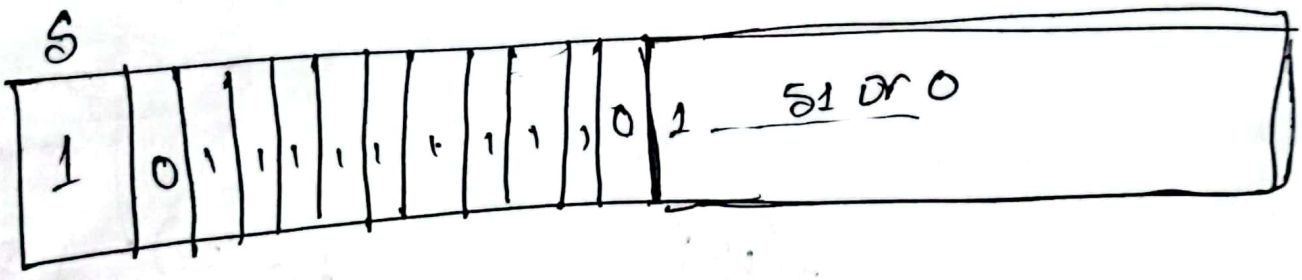
-1

-1023

So double precision representation

$$(-1)^S \times (1 + \text{Fraction(52 bit)}) \times 2^{(\text{Exponent} - \text{bias})}$$

$$= (-1)^1 \times (1 + .1 \text{ --- } 51 \text{ zero}) \times 2^{(1022 - 1023)}$$



1023 in binary

Converting binary to decimal.

(A) binary dec

1 10000001 | 01000000000000000000000
sign | exponent | fraction (23)

1 2 3 4 5 6 ... 23

$2^{0} \times 1 + 2^{-2} = \frac{1}{4} = 0.25$

So sign = 1 Exponent 10000001 = 129 Fraction = 0.25
(128 — 1)

$$= (-1)^{S} + \left(1 + \text{Fraction}\right) \times 2^{(\text{Exponent} - \text{bias})}$$

$$= (-1)^{1} + (1 + 0.25) \times 2^{(129 - 127)}$$

32 bias, 127

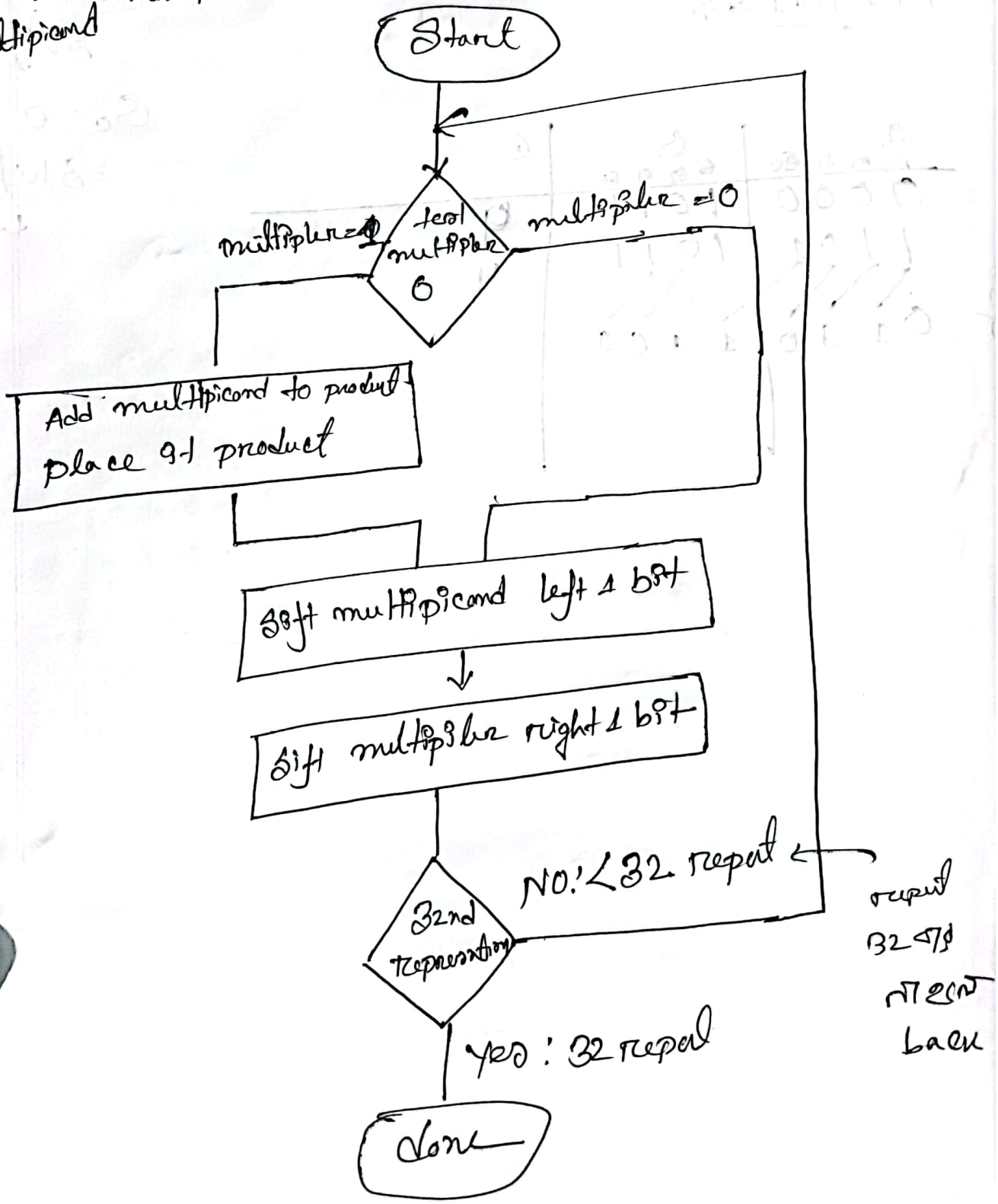
$$= -1 + 1.25 \times 4$$

$$= (-5.0)_{10}$$ Ans

1st multiplication Algorithm

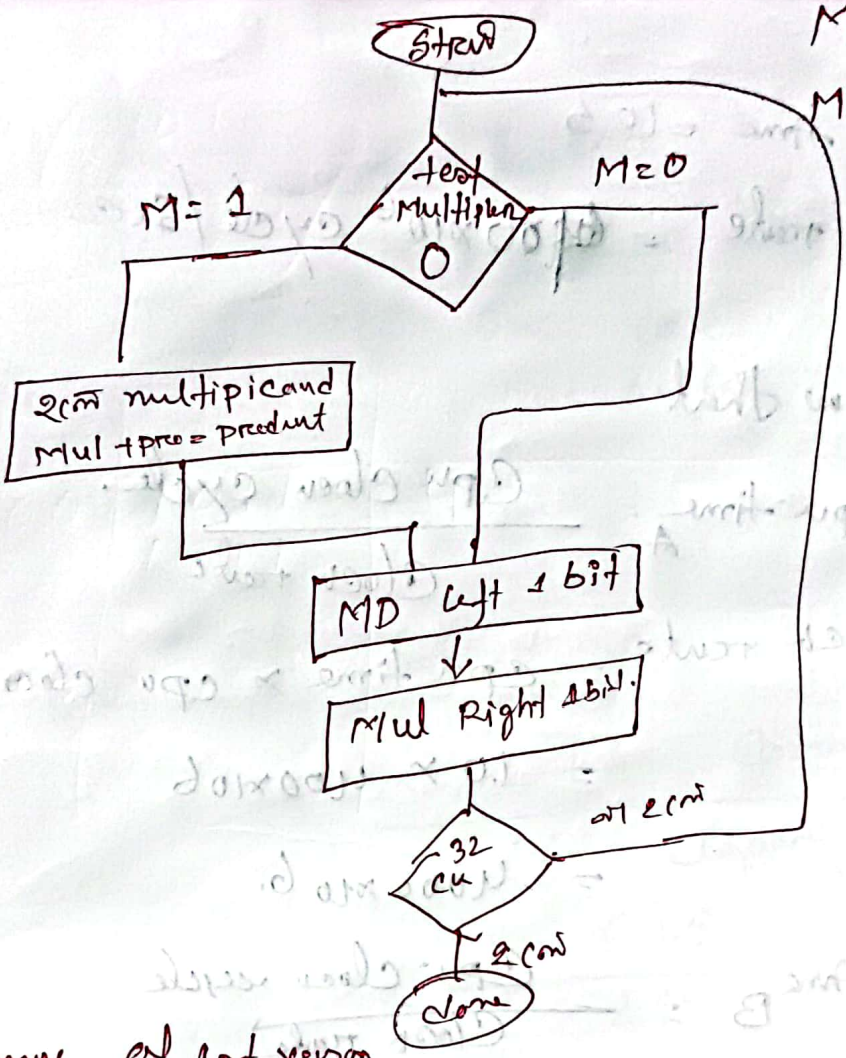
$3 \times 4 = 12$ ← product

multiplicand / multiplier

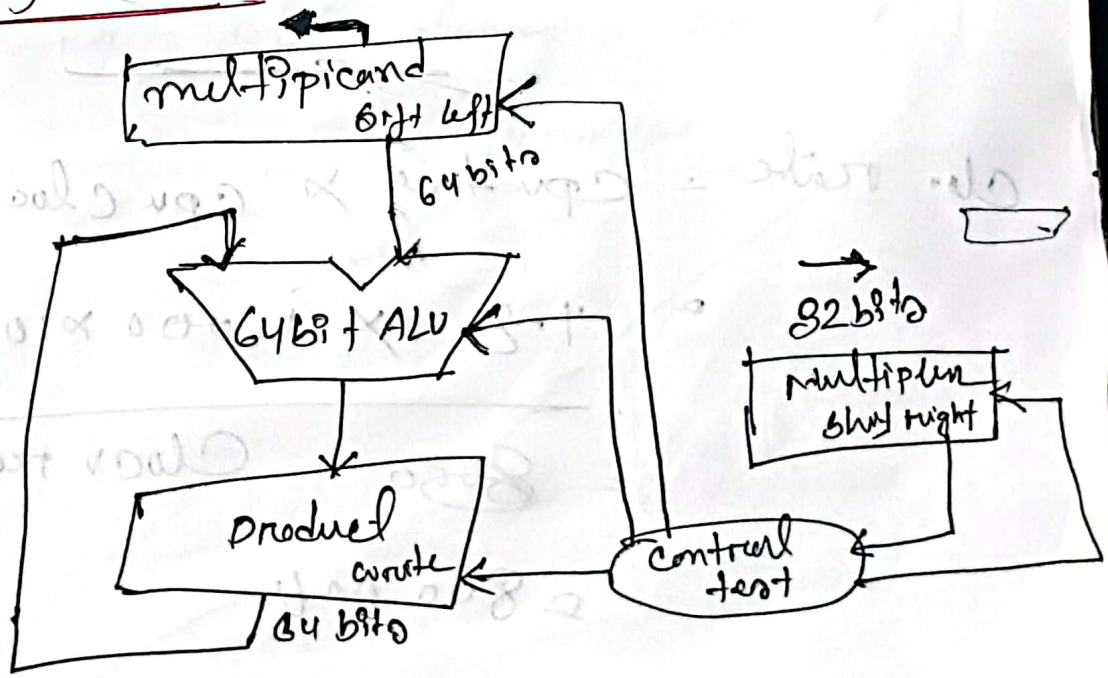


repeat 32 বার না হলে back

Multiplier = Mul
Multiplicand = MD



hardware of 1st version

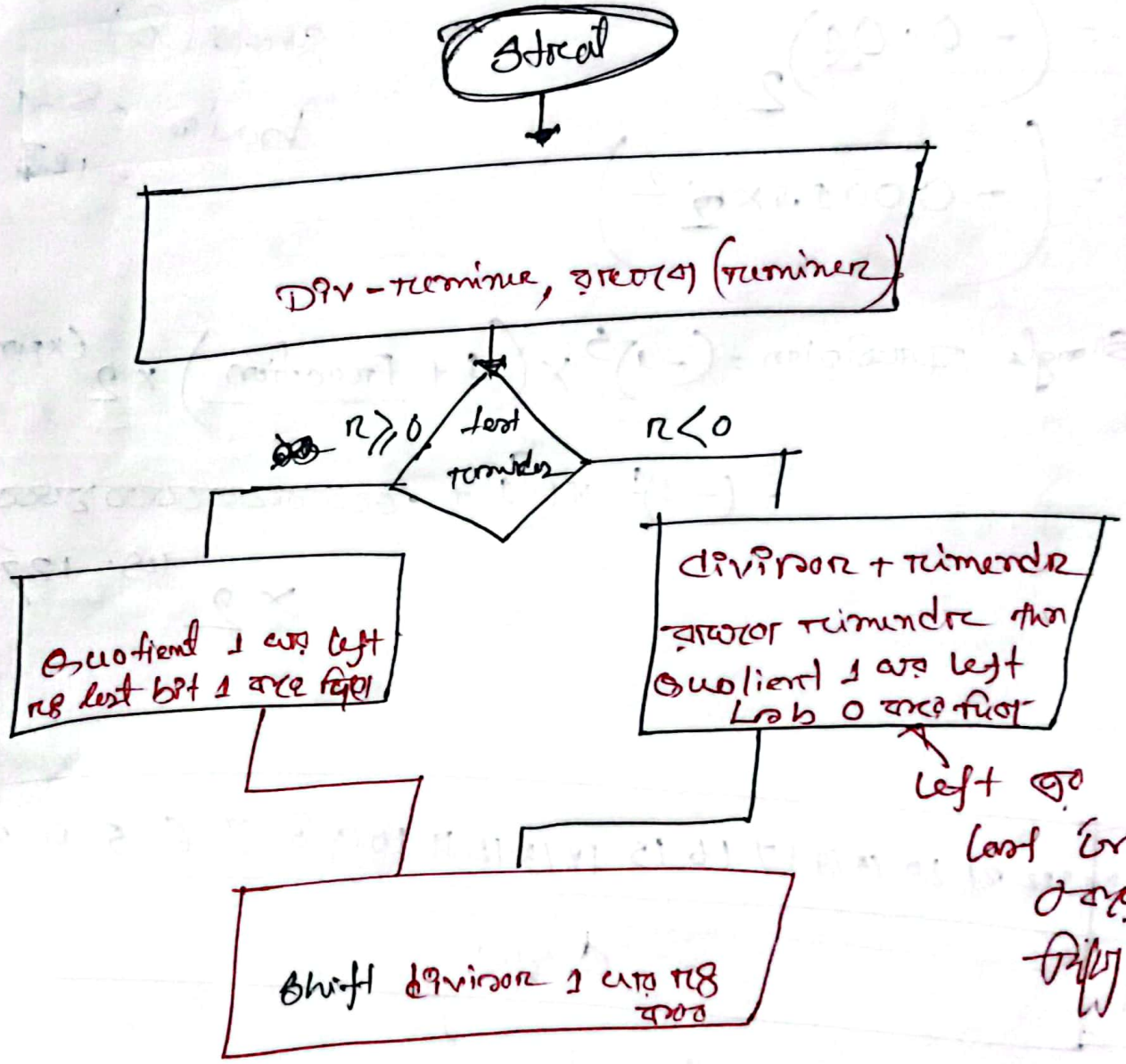


Restoring division Algorithm

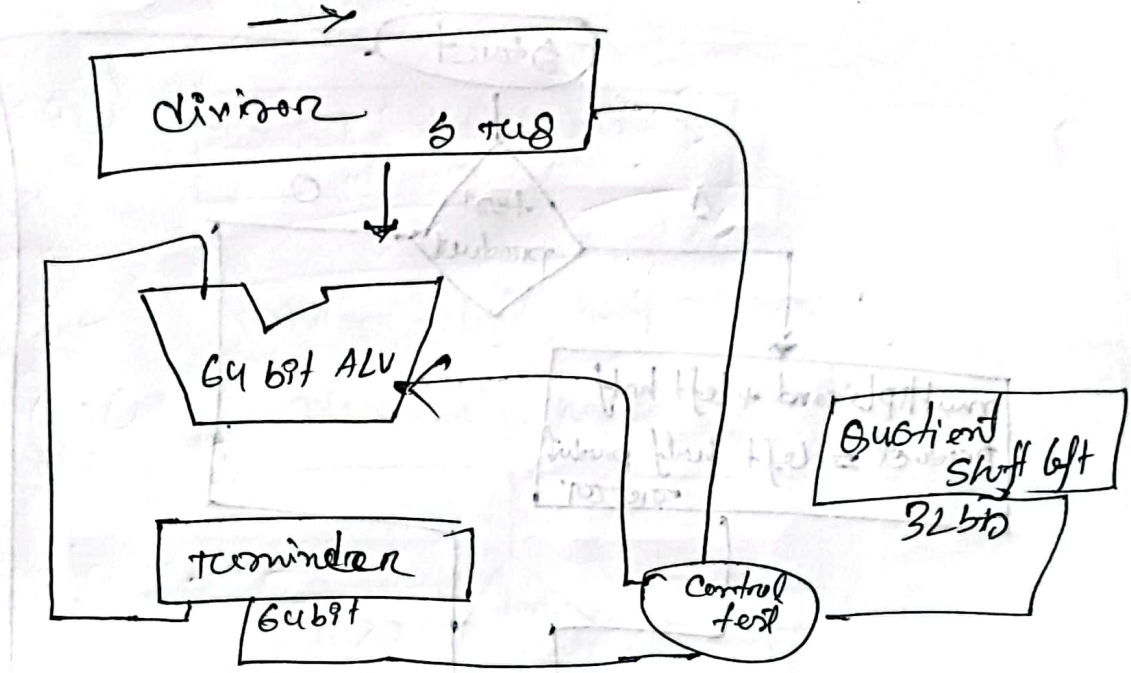
divisor
(
8 | 32 | 4 ← quotient
    32
    ---
    0 ← remainder

divided

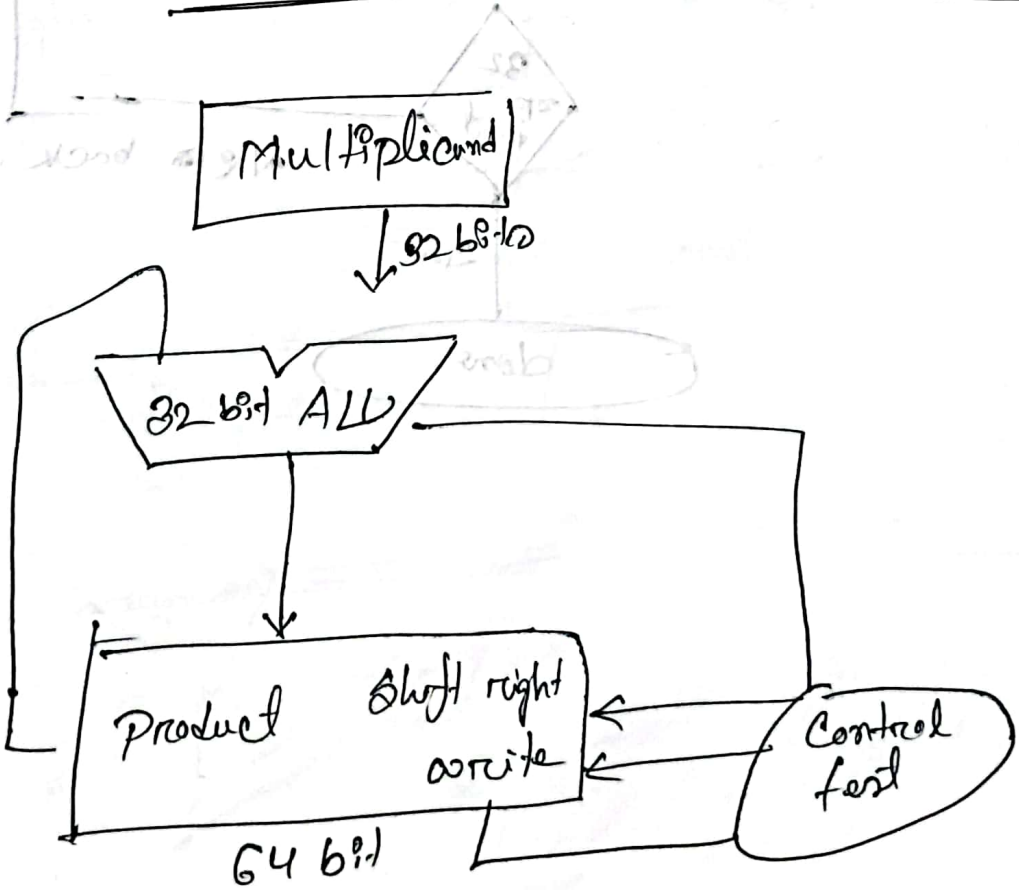
Flow Chart



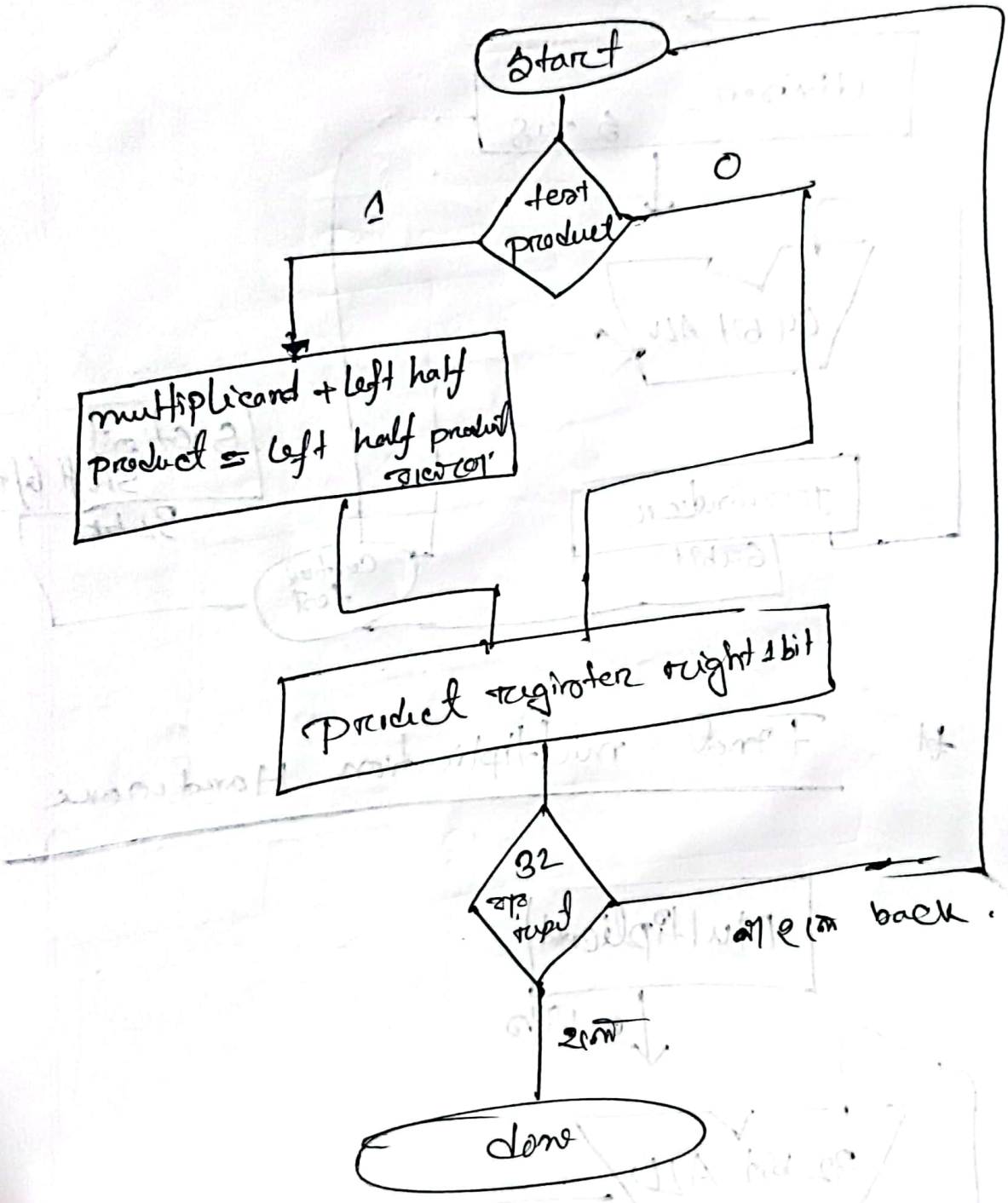
Int division hardware



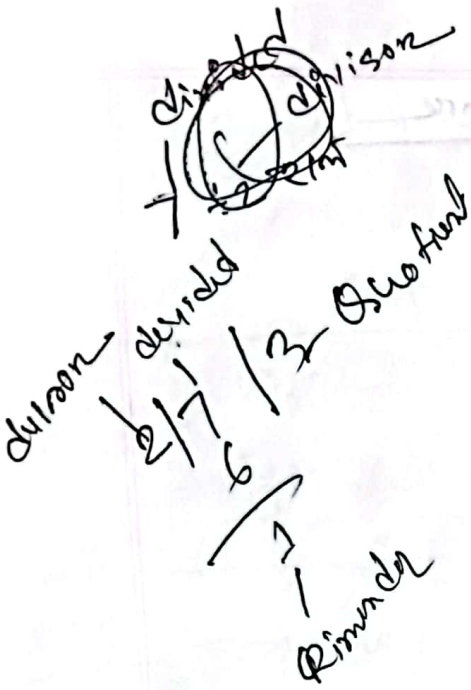
Final multiplication Hardware



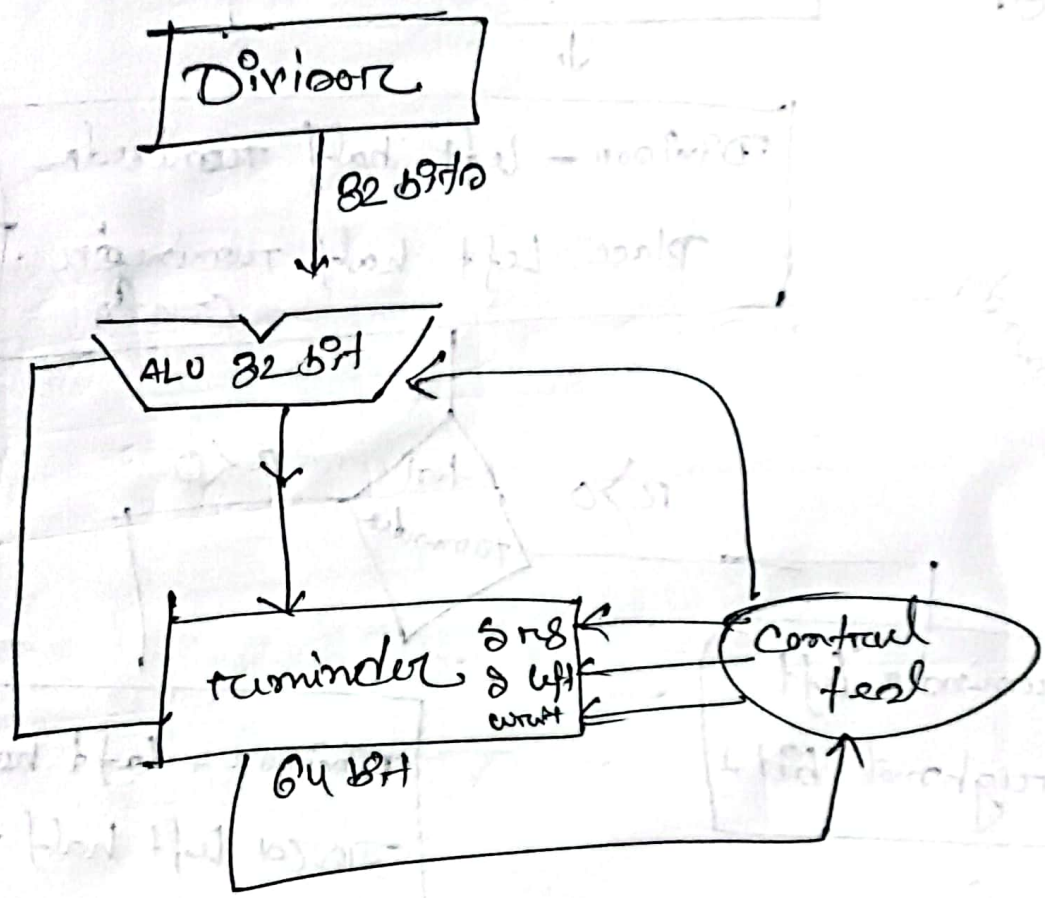
Final version of multiplication



Final version of Division



Final version hardcoare



division Final version Algorithm

Math

$(2 \div 7)$

Step	Divisor	remainder / divided
Initial	0010	0000 0111
Shift left 1 (reminder)	0010	0000 1110
	0010	1110 1110
rem = rem − div	0010	0000 1110
rem + div = rem	0010	0001 1100
Shift left	0010	1111 1100
rem = rem − div	0010	0001 1100
rem + div = rem	0010	0011 1001
Shift left	0010	0001 1001
rem = rem − div	0010	0011 0010

0 হলে + না হলে left shift করে শেষে 0 বসাতে হবে

SL left

0 হলে 1 হবে

Side calculations:

0000 − 0010 = 1110

1110 + 0010 = 10000

0001 − 0010 = 1111

1111 + 0010 = 10001

0011 − 0010 = 0001

0001 1000

0011 0001

(i) Sub $s2, $t5, 20 (R formed)

Sub rd, rt, shift

	OP	rs	rt	rd	shamt	funct
d →	0	18	20	18	20	34
b →	000000	10010	10100	10010	10100	100010

01101

(ii) bne $s3, $1, 100 (I formed)

OP rs rt Immediate

	OP	rs	rt	Immediate
d	5	$s3/19	$1/17	0023
b →	000011	10011	10001	0000000000100011

(iii) Sw $s7, 80($s2) (I format)

OP rt Immediate rs

(80+s2) = 37

primary

	OP	rs	rt	Immediate
d	43	$s2/18	$s7/23	80
B	101011	10010	10111	0000 0000 0101 0000