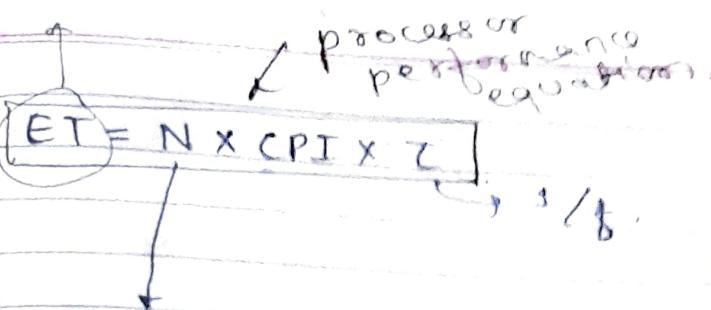


execution time

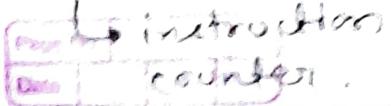


IC instructions

$$\text{MIPS Rate} = I_c = \frac{f}{T \times 10^6} = \frac{f \times I_c}{f \times 10^6}$$

start
stop
total no. of cycles

$$cc = I_c \times CPI$$



$$SAC = \frac{\text{Instr.}}{\text{Program}} \times \frac{cc}{\text{Program instr.}}$$

- Q. Consider the execution of an object code with 2×10^6 instruction on a 400MHz processor. The program consists of 4 major types of instruction. The instruction mix & no. of cycles (CPI) are as shown in table given below based on result of program based experiments. calculate avg. CPI for implementation & calculate corresponding MIPS rate.
- Ans. cycle per instruction.

IT	<u>cache hit</u>	CPI
A & L	↑	1
L & S	(CH)	2
Br.		4
m R	(M)	8

cache miss.

$$\begin{aligned}
 &= 0.6 \times 1 + 0.8 \times 2 + 0.12 \times 4 + 0.1 \times 2 \\
 &= 1.4 + 1.6 + 0.48 \\
 &= \frac{3.48}{1.62} \approx 2.24 \text{ cycle/instruction}
 \end{aligned}$$

CPI $\rightarrow 2.24$

72.

$$\begin{aligned}
 \text{MIPS rate} &= \frac{400 \text{ MHz} \times 10^6}{2.24 \times 10^6} = 181.8 \text{ million instruction/sec.}
 \end{aligned}$$

$$ET = (2 \times 10^6) \times (2.24) \times \left(\frac{1}{400 \times 10^6} \right)$$

$$= \frac{1.12}{400} = 0.0112 \text{ seconds}$$

Q. ET (in seconds) of 4 programs on 3 computers are given below.

P↓	3 comp. ET (in secs)		
	CA	CB	CC
P1	1	10	20
P2	1000	100	20
P3	500	1000	50
P4	100	800	100

Assume that 10^9 instr. os are executed by each of the program. Calculate MIPS of each program on each of 3 m/c. Based on these ratings can you draw clear conclusion regarding their ~~regarding~~ relative performance.

- Q. Consider 3 process executing the same instruc^a set
- P1 has clock rate 3 GHz & CPI 1.5,
 P2 has clock rate 2.5 GHz & CPI 1,
 P2 " " " 4 GHz & CPI 2.2.
- (a) Which processor has highest performance expressed in instr./seconds?
- (b) if the pro. each execute a program in 10 sec find no. of cycle be no. of instructions?
- (c) We ~~are~~ are trying to reduce execut^a time by 30% while compromising CPI incr. by 20%. What clock rate should hv to have this reduction?

(a) P1 \rightarrow MIPS = $\frac{3 \times 10^9}{1.5} = 2 \times 10^9$ instr./seconds

P2 \rightarrow $= \frac{2.5 \times 10^9}{1} = 2.5 \times 10^9$

P3 \rightarrow $= \frac{4 \times 10^9}{2.2} = 1.8 \times 10^9$

Vence, P2 is having highest performance in case of instr./seconds.

(b) P1 \rightarrow cycles. $\rightarrow 10 \times 1.5 \times 10^9$.

} Cycles

P2 $\rightarrow 10 \times 2.5 \times 10^9$

P3 $\rightarrow 10 \times 4 \times 10^9$

$\frac{50 \times 10^9}{2.2}$

100% + 100%

Page No.	
Date	

$$P_2 = \frac{15 \times 10^9}{1.5} = 10^{10}$$

$$P_2 = \frac{2.5 \times 10^9}{2.2} = 2.5 \times 10^9$$

$$P_3 = \frac{4.0 \times 10^9}{2.2} = 1.8 \times 10^{10}$$

c)

(i) $P_f = E_T \times f = \frac{N \times CPT}{E_T}$

$$f = 1.5 \times$$

$$\begin{aligned} CPT_n &= CPT_0 \times 1.2 + CPT_0 \times 0.03 \\ &= CPT_0 \times (1.02) \end{aligned}$$

$E_T \rightarrow 10 \text{ sec} \rightarrow 30\% \text{ redo} \Rightarrow E_{T_n} = E_{T_0}$

$$f = \frac{N \times CPT}{E_T} = 10^{10} \times 1.8$$

RISC - V \rightarrow little endian (least significant of lower address)

Register features to access (or bind) at r_0 $\&$ r_1

Register Translators Store C

RISC - V

dest \downarrow
~~RS, R2, R3~~ \rightarrow 2 operands.

3 ADD \downarrow \downarrow \downarrow \rightarrow $RS \rightarrow RS + R3$

2 ADD $RS, R2 \rightarrow RS \rightarrow RS + R2$

1 ADD RS

0 ADD

$$Z = K + (B * C)$$

3 operand instruction

MUL RS, B, C

ADD Z, K, B*C.

ADD Z, K, RS

2 operands

MUL RT,

MUL RS, B, Z, C

MUL RS, C, Z, B

ADD RS, Z, B

ADD RS, Z, C

1 operands

MOV AX, K

MOV AX, C

STOS ~~MOV~~ Y, AX

MUL B

(AX \rightarrow B*C)

add K

(AX \rightarrow K+B*C)

St Z. (Z \rightarrow AX)

• n. O operands -

$$Z = K + B * C.$$

prefix K B C * + .

Push K

Push B

Push C

MUL

→ POPS C & B, multiplies & again push on stack

ADD

→ POPS B*C & A, adds, push on stack

POP Z.

ex. of little endian & big endian Architecture

IS-A's

→ x86

RISC & CISC

MIPS

ARM

RISC-V

SPARC

instruction set architecture

Stage 1.

[inst. fetch.]

Stage 2

[src. registers]

Stage 3

[ALU]

Stage 4

[memory Access.]

Stage 5

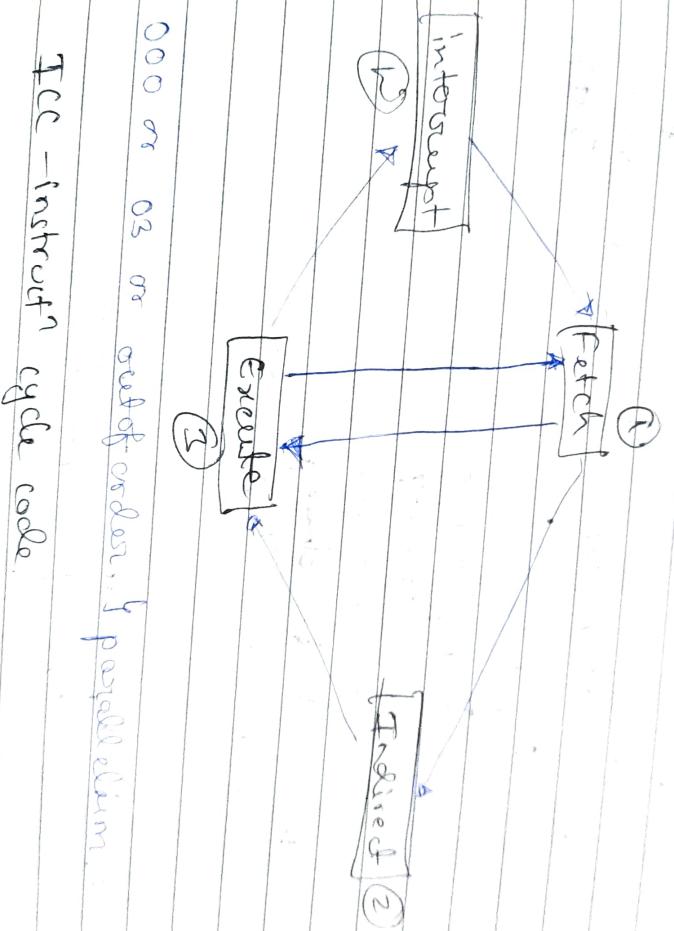
[Destination register]

→ Both no action for branching instructions

Page No.

- Add P₃, R₅, 25%
heat until all
IR - monomer
Decade []
3. Reheat []
4. Reheat []
R₃ & R₅

Chpt. 14th as 20th starting
Chpt. 8th Rak's



000 00 03 33 0 0 0 0

TCC-instruction cycle code.

complex instruction set computer (CISC) vs reduced instruction set computer (RISC)



control signal
MC - memory function complete

128 pg. no.

S0S86 → pentium. By CISC → variable length instr.

RISC-I to RISC-V RISC → for length instr.

handwired architecture.
can't be altered.

CISC

instr.

If to be in WB

maybe more than
5 steps.

more registers are

used so steps
are enough.

It can't be changed
as hardware implemented
is done.

→ CISC used!!

* microprogram controller

2 steps → requesting. (1)
→ execution. (2).

Static linking → compile time
Dynamic linking → run time

Pno. 1
Date

CISC

- A less int. are need, less program space.
- $\times 86$
- variable length
- may not fit in single word.

RISC

- compacted.

PC - I to PC - II

- fixed length

instr. stored in words (32 bit).

ADD R₁, R₂

ADD R₃, R₁, R₂

fixed 3 step

execution.

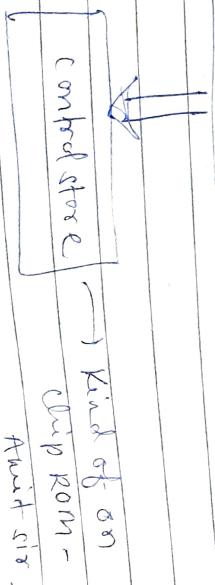
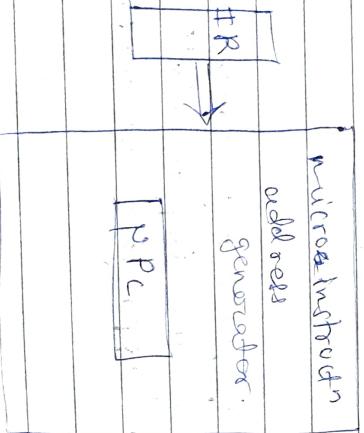
- can be altered:

• faster, but can't be changed

- microprogram controller
- hardware implementation

* Microprogram Controller - 185 pg.no. (5.2.2)

PC → μPC

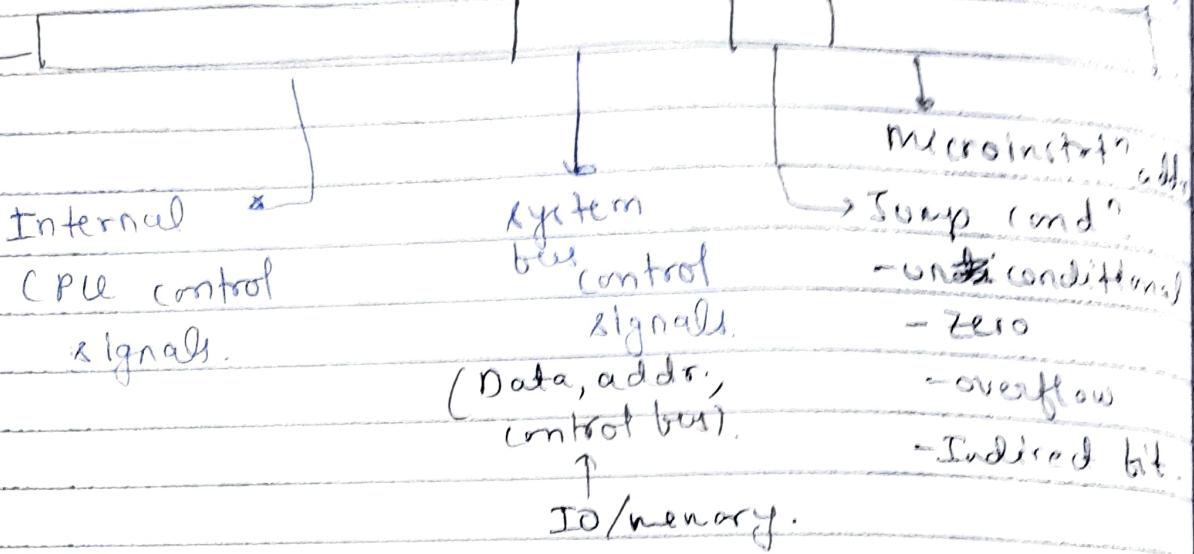


2 types of microinstructions

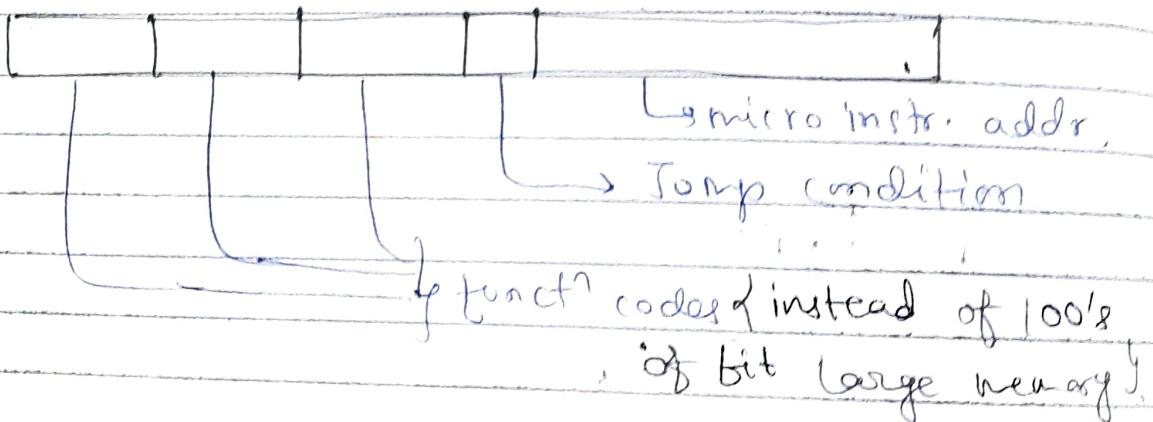


vert. microinstructions.

→ Drawback → more space required!! hence cost increases



vertical microinstr.



• 2 ways of generating control signals

→ Hardware implementation

→ microprogrammed controller

control signal generat?

i) Hardware (RISC)

ii) Microprogram controller (CISC) ^{on chip}
separately from main memory.

It has control word memory i.e.

* Control memory is usually implemented using FLASH ROM as it is writable yet non volatile

Page No.	
Date	

- On the pentium 4, machine instructions are converted into a RISC-like format, most of which are executed without the use of microprogramming.
- Microprogram or firmware is somewhere midway between hardware & software.
- It is easier to design firmware than hardware but it is more difficult to write a firmware program than a software program.

coz 2 memories the microprogrammed control has to deal & main memory is control memory
 ↴ approach is below

on
off-chip!!

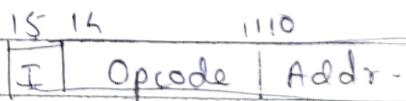
7 bit addr.

$$\text{control memory} = 2^7 = 128 \text{ bytes}$$

↓
first 64 i.e. 0-63 reserved

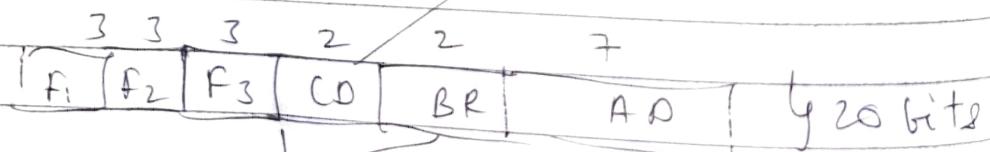
used by or
occupied by 16 instr.

* Computer instr. format.



0 64, 65, 66 are
used for 3
microinstructions
fetch routine
needs 3 microinstr.

ADD	0000
BRANCH	0001
STORE	0010
EXCHANGE	0011



microinstruction code format

branch field

SBR → subroutine.

Page No.		
Date		

F1

000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLR AC
011	$AC \leftarrow AC + f$	INC AC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0-10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

F2

NOP
SUB
OR
AND
READ
ACTDR
INCDR
PCTDR

F3

NOP
XOR
COM
SHL
INCPCL
PCLAR
Recovered

CD

DO

O1

10

11 $AC = 0$ → nothing gone to AC, gone to memory

Morris Manoto - Unit - 7 chpts
Cir will give!
Page No. _____
Date _____

BR

00 JMP
01 CALL
10 RET
11 MAP

64, 65, 66.

→ make all 7 bits of AD
to 0!! Again starts
from new, next instruction

FETCH:

OAG 64

PCTAR

READ, INCPC

DRTAR

U U U

JMP JMP MAP

NEXT NEXT

INPRCT:

- READ

DRTAR

U U

JMP

RET

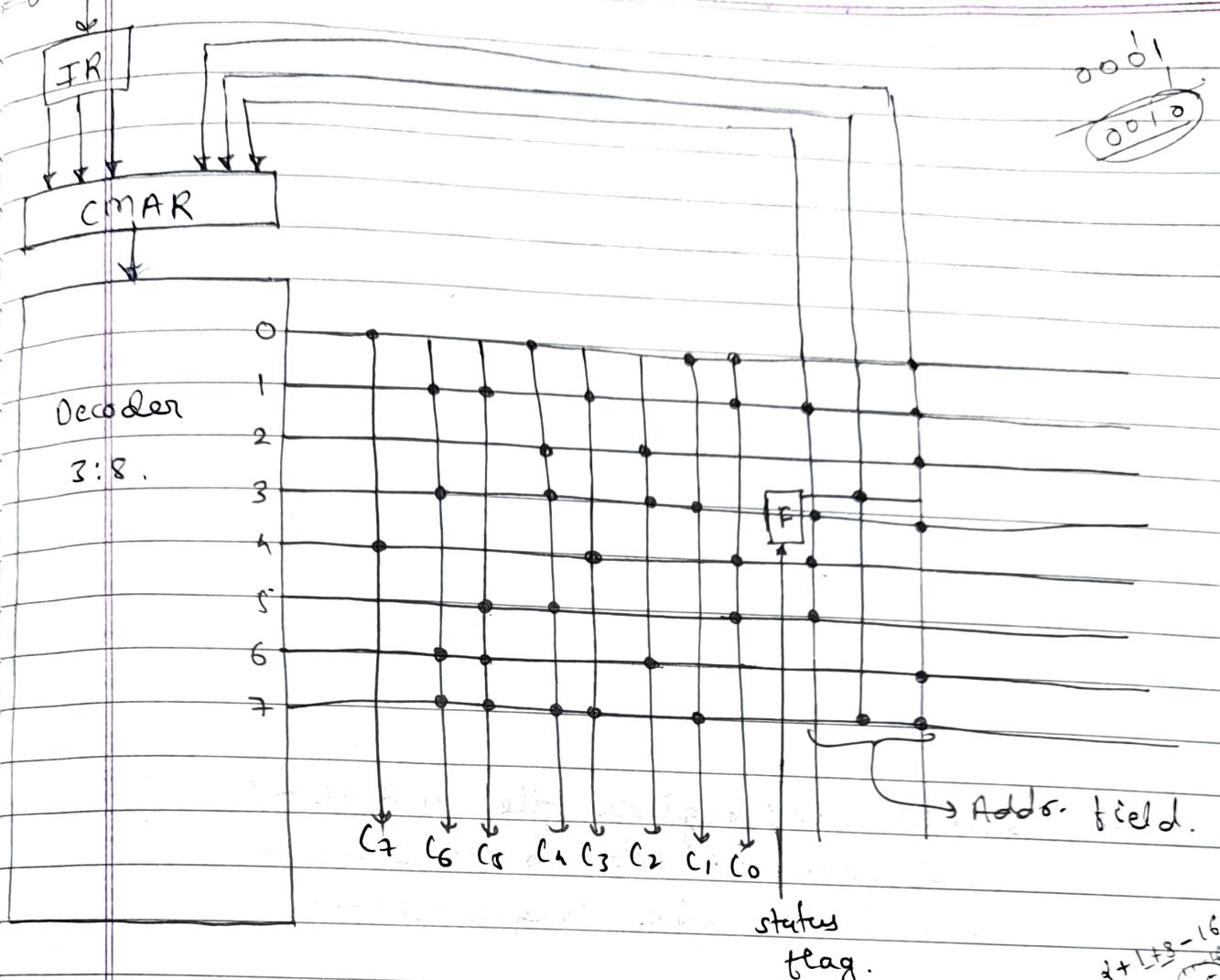
NEXT

→ returning the addr.

instr. fetched
from main memory.

* Sign extension?

Page No.		
Date		



each row is a microinstruct^.

2+158-16
15

* Overflow rule - If 2 no.'s with the same sign (both +ve or both -ve) are added, then overflow occurs if and only if the result has the opp. sign.

* Subtraction rule - To subtract one no. (subtrahend) from another (minuend), take the 2's complement (negation) of the subtrahend and add it to the minuend.

RISC-V Manual

PLC18 → 4000 register
ARM → 32 reg file

Par. No.
Date:

CISC

- complex instrudn
- art computer
- flexible instr. format
- more addressing modes
- memory based
- (load, store)
- less registers
- poor pipelining
- variable execution time
- easier to program
- microprogrammed CU
- SOR5, R086, MP

RISC

- Reduced instruction set computers
- rigid instr. format
- less addr. nodes
- registers based
- more registers
- excellent pipelining
- fixed execution time
- more complex
- hardware CU
- microcontroller, ARM

PLC

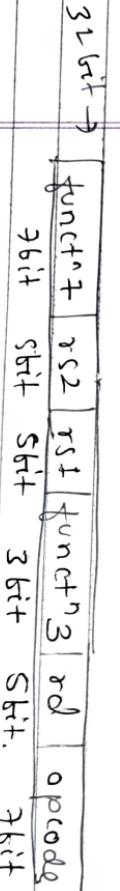


→ 32x64 bit register file → RISC-V
Doubleword = 64 bit, word = 32 bit.
each instr. fix length - 32 bit

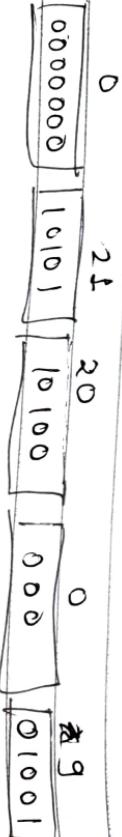
- R-format. Register-Register

Add R₄, R₅, R₃

R₇ ← R₄ + R₃



add x9, x20, x21 x9 ← x20 + x21



opcode 51

1 word = 4 bytes = 32 bits

R, I, S, U, SB, CJ
 add, xor, mul
 add i, lw, jalr, slli
 12 bit I

beq, bge

Page No.

Date

I	imm[11:0]	rs1	3 bit funct ⁿ 3	5 bit rd	7 bit opcode
S	imm[11:5] (16 bit)	rs2(5 bit)	funct ⁿ 3	imm[4:0]	opcode
SB	funct ⁿ 6	immed (6 bit)	rs1	funct ⁿ 3	rd
U					shift
UJ					

beq : rs1, rs2, LI

if (rs1 == rs2) branch to instr. Labelled L

bne

not equal!

185, 122

715

188

128

057

32

12875

16

09

8

1

1 0 1 1 1 0 0 1

- 00

185

00111010

+ 00

01112

64

58

32

26

10

10

D 1 0 1 1 0 0 1 1

- 71

- 128 + 32 + 16 + 8 + 1

32

16

8

71

128

87

71

-2^{n-1} to $+2^{n-1} - 1$

2nd unit pattern

Page No.	
Date	

$$1) 6 + 13 = 19$$

$$2) -6 + 13 = 7$$

$$3) 6 - 13 = -7$$

$$4) -6 - 13 = -19$$

$$\begin{array}{r} 11001100 \\ - 1101110 \\ \hline 00101110 \end{array} \quad \begin{array}{r} 11000011 \\ - 11101000 \\ \hline 11011010 \end{array} \quad \begin{array}{r} 11000011 \\ - 00011000 \\ \hline 11011011 \end{array}$$

$$\begin{array}{r} 111000 \\ - 10011 \\ \hline 000101 \end{array}$$

$$\begin{array}{r} 111100001111 \\ - 11001110011 \\ \hline 1100 \end{array}$$

$$\begin{array}{r} 00000110 \\ + 00001101 \\ \hline 00010011 \end{array}$$

$$\begin{array}{r} 111110\cancel{0}10 \\ - 00001101 \\ \hline 10000011 = +7 \end{array}$$

$$\begin{array}{r} 001100 \\ - 001101 \\ \hline 00 \end{array}$$

$$\begin{array}{r} 00000110 \\ 11110011 \\ \hline 11111001 \end{array}$$

$$\begin{array}{r} 11111010 \\ 11110011 \\ \hline 11101101 \end{array}$$

$$11110010$$

$$\begin{array}{r} 11111001 \\ - 0000101 \\ \hline 11110010 \end{array}$$

$$\begin{array}{r} 11101101 \\ - 11101101 \\ \hline 00000000 \end{array}$$

$$+ 2 + 1$$

$$7 \neq$$

$$\begin{array}{r} 111000 \rightarrow 111000 \\ 110011 001101 \\ \hline 1000101 \end{array}$$

$$1 + 8 + 16 + 32$$

$$+ 64$$

$$\cancel{128}$$

$$64$$

$$32$$

$$16$$

$$9$$

$$121$$

$$\begin{array}{r} 111100001111 \\ 001100001100 \\ \hline 100010001100 \end{array}$$

$$+ 4 + 8 + 32$$

$$96 + 64$$

$$13$$

$$128 - 128$$

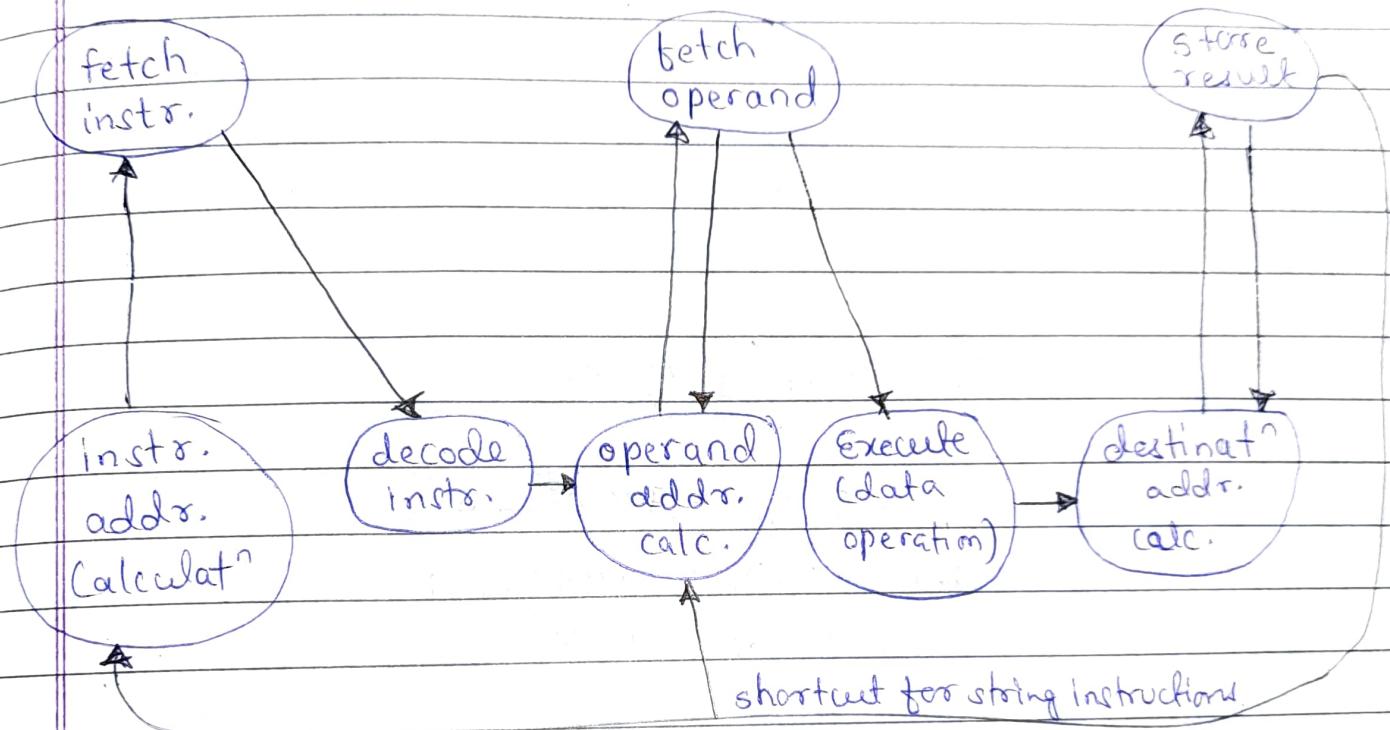
$$129 - 129$$

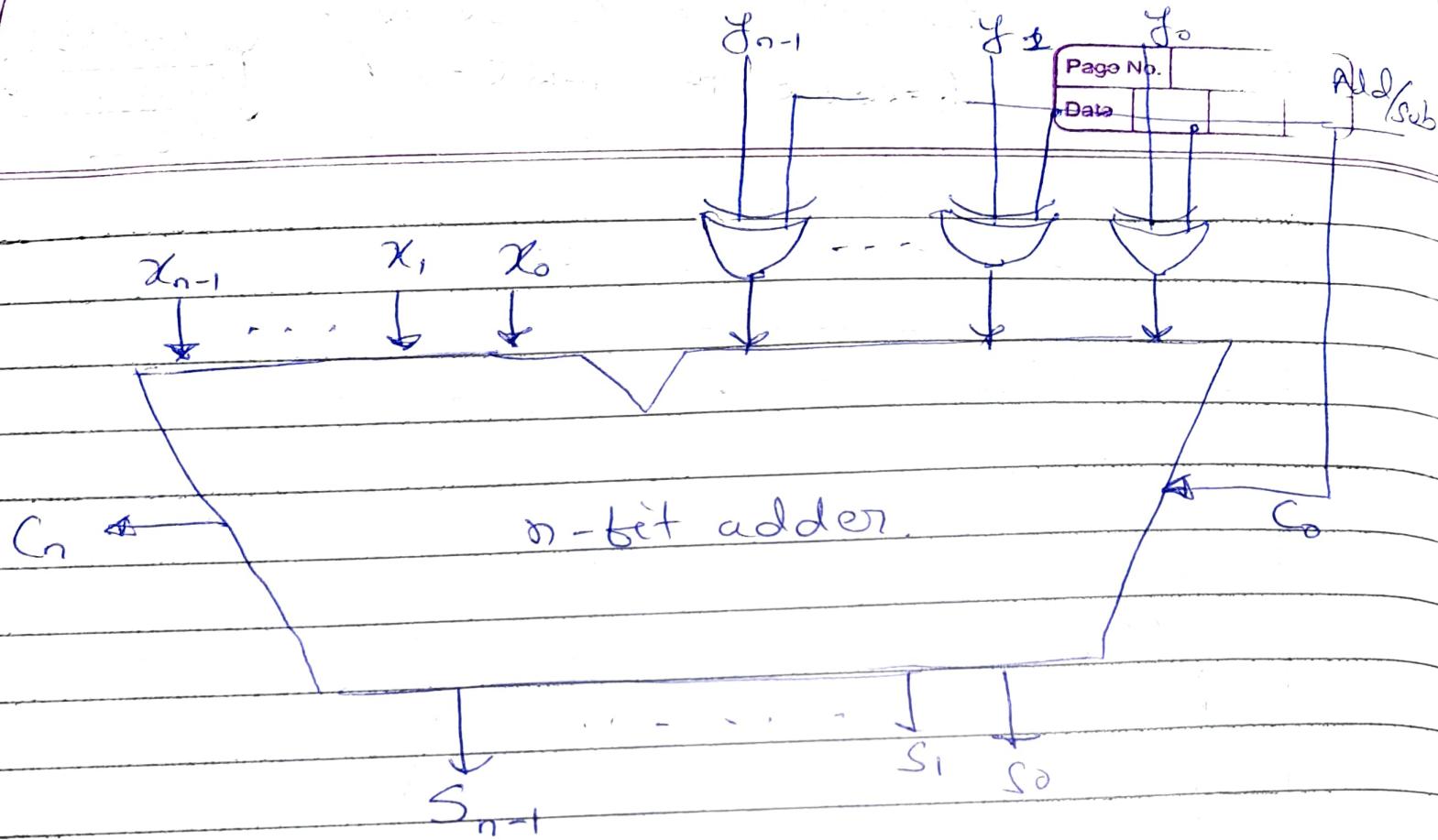
RISC-V to C code / fERE!!
C-code to RISC-V

Page No.	
Date	

- find the following differences using 2's complement arithmetic.

- Instruction cycle state transition diagram-





$$S_i = x_i \oplus y_i \oplus C_i$$

$$C_{i+1} = x_i y_i + x_i C_i + y_i C_i$$

$$C_{i+1} = x_i y_i + (x_i + y_i) C_i$$

$$C_{i+1} = G_i + P_i C_i$$

$$G_i = x_i y_i$$

$$C_i = x_i + y_i$$

m.m - 16 KB $\rightarrow 2^{14}$

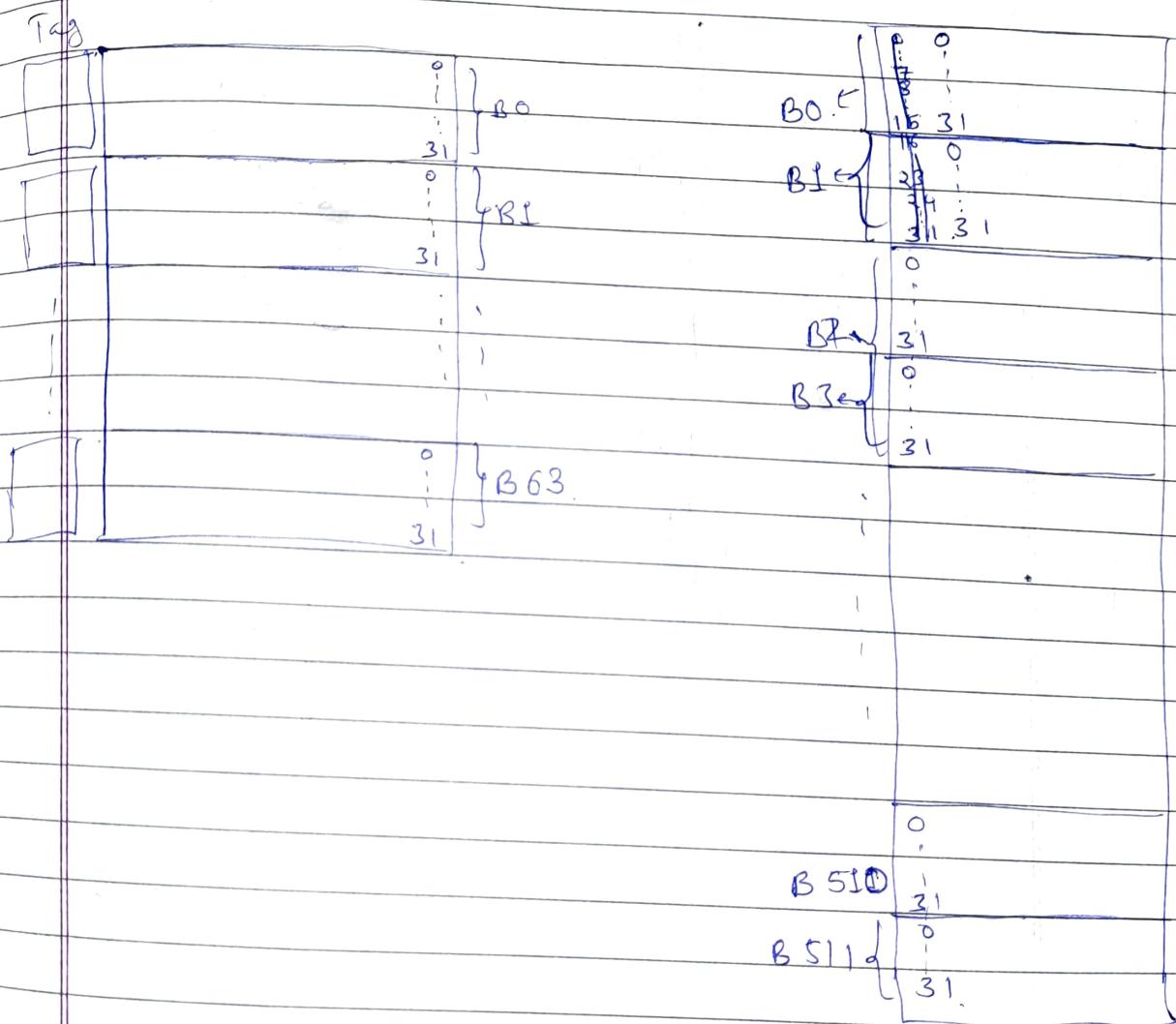
c.c - 2 KB $\rightarrow 2^9$

Block size - 32 bytes $\rightarrow 2^5$

Block c.c $\rightarrow 2^9 = 2^6 = 64$
 2^5

no. of blocks in m.m = $\frac{2^{14}}{2^5} = (2^9) = 512$

each block consist of 32 bytes



m.m address

Tag $\rightarrow (0, 511)$

Tag $\rightarrow 512 (2^9)$

Tag $\rightarrow 9$ bit.

tag
(9)
Locatⁿ/Block
(5 bit)

$9+5 \rightarrow 14$ bit.

① $m \cdot m = 16MB \rightarrow 2^{24}$

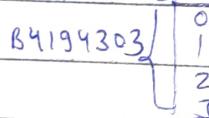
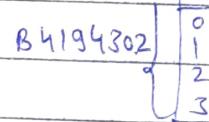
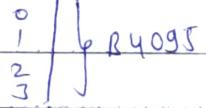
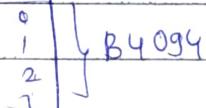
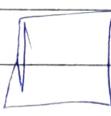
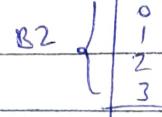
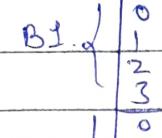
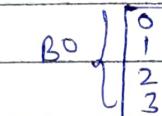
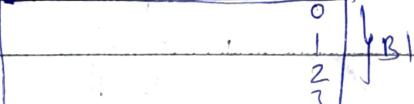
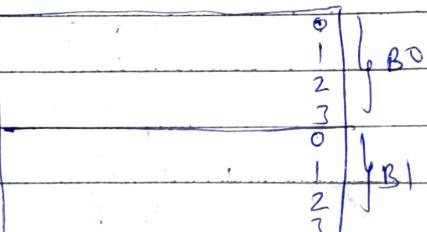
$c \cdot m = 16K \rightarrow 2^{14}$

block size - 32 bits $\rightarrow 4 \text{ bytes} \rightarrow 2^2$

no. of blocks in $m \cdot m = 2^{24}/2^2 = 2^{22} = 4194304$

no. of blocks in $c \cdot m = 2^{14}/2^2 = 2^{12} = 4096$.

tag



tag

22 bit $\xrightarrow{2K} 26 \text{ bit}$

4 byte

tag $\rightarrow (0, 4194303)$

tag $\rightarrow 4194304 - 2^{22}$

(hex)	tag	word.					
00000000	0.....0	00	13	57	92	46	
00000001	0....\$00						
3FFFFFFF	1.....1	00	24	68	24	68	
			33	33	33	33	
			11	22	33	44	

$n \bmod K$

$0 \bmod 4 \rightarrow 0$

$1 \bmod 4 \rightarrow 1$

$2 \bmod 4 \rightarrow 2$

$3 \bmod 4 \rightarrow 3$

$4 \bmod 4 \rightarrow 0$

$5 \bmod 4 \rightarrow 1$

Direct

$$m \cdot m \text{ block} = \frac{2^7}{2^2} = 2^5 \rightarrow 0, 31$$

$$m \cdot m \rightarrow 128 \text{ bytes} \rightarrow 2^7$$

$$\text{Block size} \rightarrow 32 \text{ bit} \rightarrow 4 \text{ bytes} \rightarrow 2^2.$$

$$C \cdot m \rightarrow 16 \text{ bytes} \rightarrow 2^4$$

$$\therefore C \cdot m \text{ block size} \rightarrow \frac{2^4}{2^2} = 2^2 = 4$$

Page No.	
Data	

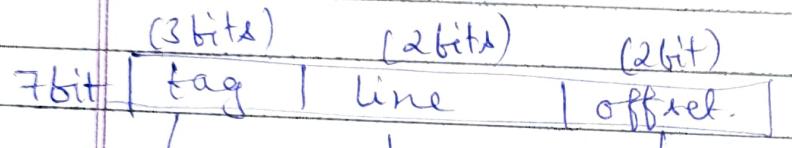
w_0	w_1	w_2	w_3	B_0
w_4	w_5	w_6	w_7	B_1

L₀ $B_0/B_1/B_2/B_3/B_4/B_5/B_6/B_7/B_8/B_9/B_{10}/B_{11}/B_{12}/B_{13}/B_{14}/B_{15}/B_{16}/B_{17}/B_{18}/B_{19}/B_{20}/B_{21}/B_{22}/B_{23}/B_{24}/B_{25}/B_{26}/B_{27}$

L₁ $B_0/B_1/B_2/B_3/B_4/B_5/B_6/B_7/B_8/B_9/B_{10}/B_{11}/B_{12}/B_{13}/B_{14}/B_{15}/B_{16}/B_{17}/B_{18}/B_{19}/B_{20}/B_{21}/B_{22}/B_{23}/B_{24}/B_{25}/B_{26}/B_{27}/w_{28}$

L₂ $B_0/B_1/B_2/B_3/B_4/B_5/B_6/B_7/B_8/B_9/B_{10}/B_{11}/B_{12}/B_{13}/B_{14}/B_{15}/B_{16}/B_{17}/B_{18}/B_{19}/B_{20}/B_{21}/B_{22}/B_{23}/B_{24}/B_{25}/B_{26}/B_{27}/B_{28}$

L₃ $B_0/B_1/B_2/B_3/B_4/B_5/B_6/B_7/B_8/B_9/B_{10}/B_{11}/B_{12}/B_{13}/B_{14}/B_{15}/B_{16}/B_{17}/B_{18}/B_{19}/B_{20}/B_{21}/B_{22}/B_{23}/B_{24}/B_{25}/B_{26}/B_{27}/B_{28}$



32 total block = 8 tag size. 4 cache line 4 bytes in a word

4 cache line

$\rightarrow 2^3$ $\rightarrow 2^2$ $\rightarrow 2^2$

$\rightarrow 2^{10}$ $\rightarrow 2^{10}$ $\rightarrow 2^{10}$

$\rightarrow 3 \text{ bits}$.

$w_{125} w_{126} w_{127} w_{128}$ B_{31}

min. 128 bytes

blk

L ₀	I block	34 words
L ₁	I block	34 words
L ₂	I block	34 words
L ₃	I block	16 words

$$= \frac{16 \text{ bytes}}{4 \text{ bytes}}$$

$$= 4$$

$$m \cdot m \rightarrow 16 \text{ MB} \rightarrow 2^{24}$$

block size \rightarrow 32 bits \rightarrow 4 bytes $\rightarrow 2^2$.

$$m \cdot m \text{ block no. } \rightarrow \frac{2^{24}}{2^2} \rightarrow 2^{22}.$$

$$C \cdot M \rightarrow 2^{14}$$

$$C \cdot M \text{ block no. } \rightarrow \frac{2^{14}}{2^2} \rightarrow 2^{12}.$$



$$\text{tag size} = \frac{2^{22}}{2^{14}} = 2^8$$

8 bits, 10 bits



24 bits

12 bits

26 bits

B_{2-1}^{22}

offset

\hookrightarrow n words

$\rightarrow 2^2 \rightarrow 2 \text{ bits}$

$$C \cdot M \rightarrow 2^{14}$$

$$\text{lines} \rightarrow \frac{2^{14}}{2^2} = 2^{12}$$

$$= 2^{22}$$

Direct:

$$\text{main memory} = 16 \text{ MB} = 2^{24}$$

$$\text{cache memory} = 16 \text{ KB} = 2^{14}$$

$$\text{block size} = 32 \text{ bits} = 4 \text{ bytes} = 2^2$$

$$\text{no. of blocks in main memory} = \frac{2^{24}}{2^2} = 2^{22}$$

$$\text{cache lines} = \text{no. of blocks in cache memory} = \frac{2^{14}}{2^2} = 2^{12}$$

