7-18

Date ___/___/_____

Advanced Computer Architecture (DA7020)

History of Electronics Computer

Designed by Univ. of Pennsylvania, USA, for US military. (around 1926)

2 People developed,
Vacuum tube technology used.

1st elec. comp. was named ENIAC (Electronic Numerical Integrator and Computer) designed by Eckhert + Manchley.

Transister invented 1947 worked as switch replaced vacuum tube.

Size small, less consumption.

Bell
AT&T Lab

→ William Shockley
→ Walter Brittmen
→ John Bardeen

} Nobel prize for Physics in 1956.

IC introduced 1958 Jack Kilby in Texas Instrument invented Integrated Circuit Chip

Page No.

Date __/__/____

→ Small Scale Integration SSI
MSI
LSI
VLSI (millions of transisters in same chip)

1960's → Super Computer - Main Frame

1970's → Mini Computer → Businesses & Researches used

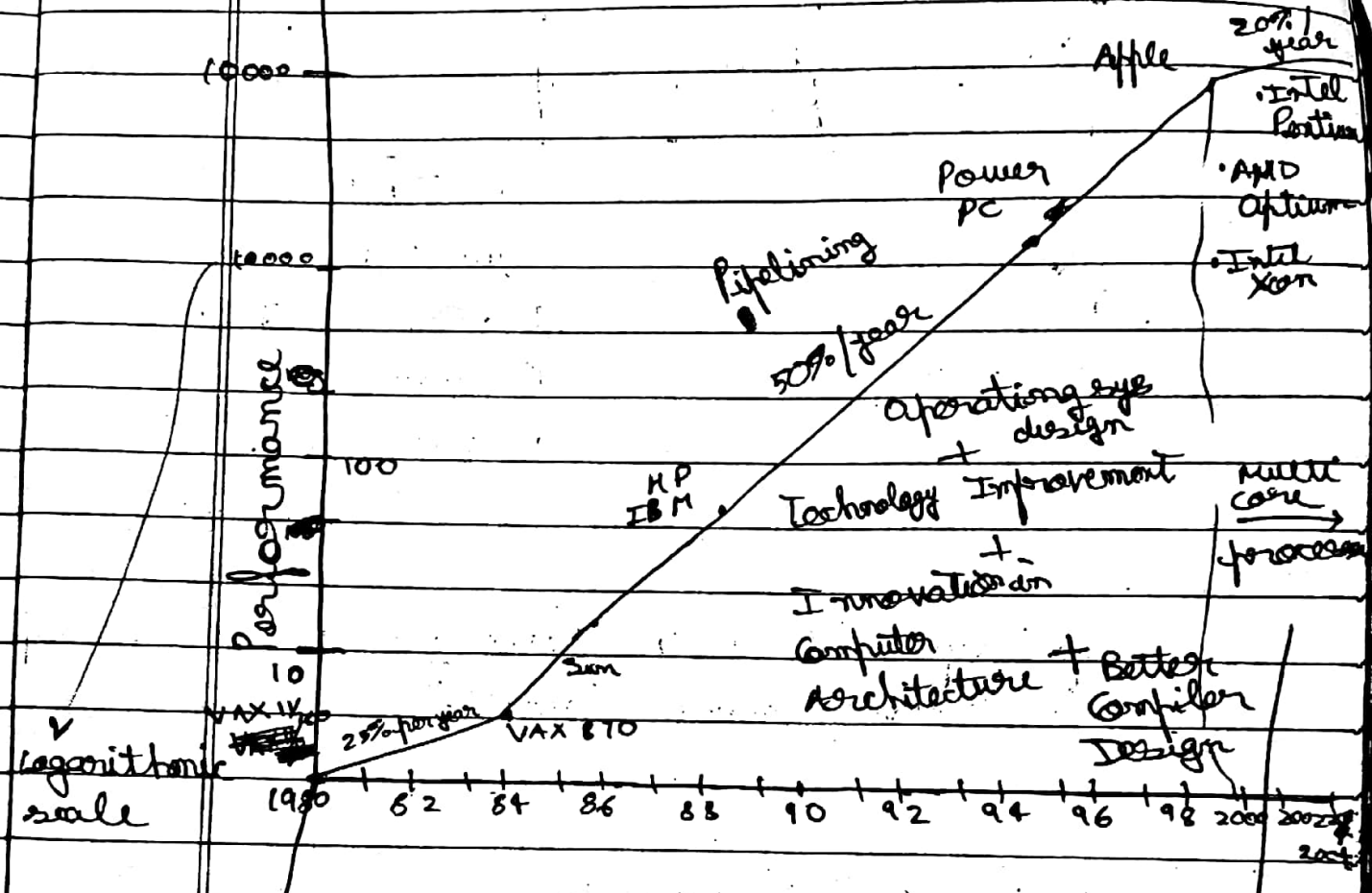
1980's → Micro Processor — Desktop Computer
→ Laptop Computer

1990's → Internet and World Wide Web, Web Services

2000's → Palm Top computer, PDA, cell phones, smart phones, Medical Diagnostic Devices, Electronic Devices

Page No.

Date ___ / ___ / ___



Logarithmic scale

VAX 11/780

Growth in Uni-Processor Performance

Gordon Moore gave
1965 Moore's law

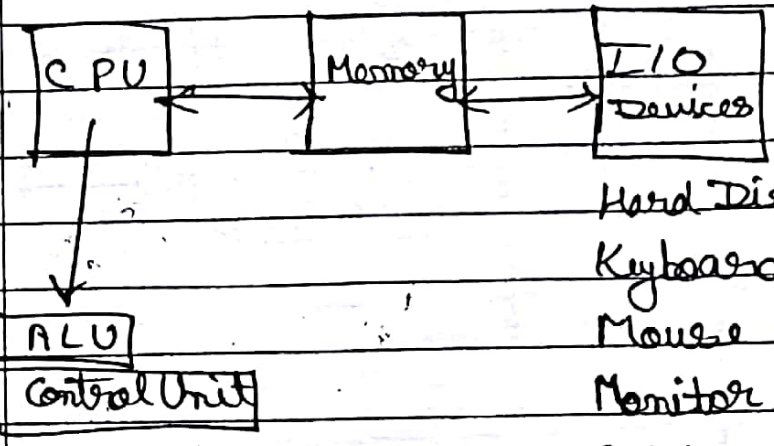
25% → due to technology
50% → innovation
20% → multiprocessor

Dual Core
Quad Core
processors

Page No.

Date __/__/____

3 Main Components of Computer



Speed gap exists b/w Memory & Processor chip
I/O devices much slower.

Computer Architecture:

→ Instruction Set Architecture (ISA).
Design

Design Principles Followed:

1) Common Case Fast. } Focus on more freq. used part
2) Regularity in Design. } Theme maintained
3) Simple Design.
4) Exploit parallelism } Multicore processors
Multiple processors

Page No. ____

Saathi

Date __/__/____

(Pipelining → Overlapping of instructions.)

5) Exploit locality of Reference } looping of instn

Speed of Development has reduced because:

→ All parallelism is exhausted.
→ With increase in no. of transistors power consumption has increased:

Trends in Power in Integrated Circuit

CMOS

Dynamic Power = $\frac{1}{2} C V^2 f$

C → Capacitive load
V → Voltage
f → Frequency of the clock

f → clock speed → if we use slower clock power consumption reduces.

C → if no. of transistors are reduced power consump. reduces.

Page No.

Saathi

Date __/__/____

V → if we reduce voltage power consumption reduces.

Static power = $I_{leakage} \times$ Voltage

= Current $\times$ Voltage
(leakage)

(When transistors are not used they still leak some current..)

Static power = 25% of Total Power Consumption.

Q) Some microprocessors today are designed to have adjustable voltage so that 15% reduction in voltage results in 15% reduction in frequency. What would be the impact on dynamic power.

$\frac{1}{2} \times C V^2 f$ → $\frac{1}{2} \times (0.85)^2 \times 0.15 f$

Page No.

Date __/__/____

$\frac{1}{2} \times C \times (1 - 0.15) V^2$

$\frac{1}{2} \times C \times (0.85)^2 \times V \times V \times (0.85) \times f$

$\frac{1}{2} \times C \times V \times V \times f$

$(0.85)^3$

$\frac{P_{old}}{P_{new}} = \frac{1}{(0.85)^3}$

$= 1.628$

$\frac{1}{2} ((.85)^2 + (0.85) = 60\%$ of original power

Page No.

2.8.18

Saathi

Date __/__/____

A summary of 3 mainstream computing classes

Features	Desktop	Server	Embedded
→ Price of System	$500 - $5000	$5000 - $5000,000	$10 - $100,000
→ Price of Microprocessor Module	$50 - $500	$200 - $0,000	$-01 - $100
→ Critical System Design Issues	Price vs. Performance, Graphic performance	Throughput, availability, scalability	Price, power consumption, application specific performance

Page No.

Date __/__/____

Throughput → work done per unit time

Architecture

Instruction Set Architecture (ISA) Design

80X86 ISA
used in → Intel Computers
CISC (Complex Instruction Computer Set architecture)

MIPS architecture
RISC (Reduced Inst. Set Computer)
embedded sys.
(Register ins^n i.e., time of execution same)
ARM processor for embedded sys. uses RISC.

Parameters for measuring computer performance:

~~Measuring and~~

Page No.

Measuring and Comparing Performance of Computers.

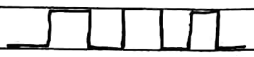
Performance $\propto \frac{1}{\text{Execution Time}}$

Execution time $= IC \times CPI_{avg} \times T$

$= \frac{IC \times CPI_{avg}}{f}$

IC → instruction count
T → Clock Period
f → Clock Frequency $= \frac{1}{T}$

CPI → Cycles per instruction



We use CPI_{avg} (Average CPI) :

because speed varies from Memory to CPU to I/O.

Page No.

Date ___/___/___

Saat

MIPS (Million Instructions Per Second)

Execution Time (secs.) $\rightarrow \frac{IC}{10^6}$

1 sec $\rightarrow \frac{IC}{10^6 \times Ex.Time}$

$= \frac{IC \times f}{10^6 \times IC \times CPI_{avg}}$

$= \frac{f}{CPI_{avg} \times 10^6}$ (MIPS Rating)

e.g;
Program = 100,000 instructions

Instruction Mix :	%	CPI
1) Arithmetic	60%	1
2) Data Transfer	20%	4
3) Control instructions	20%	2

Saathi

Date __/__/____

Find Execution time of the program if the computer clock has a frequency of 400 MHz.

A)

		CPI
$\frac{60}{100} \times 100,000 =$	60,000	→ 60,000
$\frac{20}{100} \times 100,000 =$	20,000	→ 80,000
"	20,000	→ 40,000

~~60,000 + 20,000 = 60,000 → 1.8~~

$$\frac{60,000 + 20,000 + 20,000}{100,000} = 1.8$$

$CPI_{avg} = 1.8$

$$\text{Execution time} = \frac{250 \times 1.8}{400 \times 10^6}$$

$$= \frac{1.8}{4000}$$

$$= 450\ \mu sec$$

Page No.

Date __/__/____

$$\text{MIPS rating} = \frac{400}{1.8 \times 10^6}$$

$$= \frac{400}{18 \times 10^5}$$

$$= 222.22 \times 10^{-6}$$

$$= 222.22\ \text{MIPS}$$

Scale

Memory & Storage

Decimal Term	Decimal value	Binary value
KB	10^3 B	2^{10}
MB	10^6	2^{20}
GB	10^9	2^{30}
TB	10^{12}	2^{40}
PB	10^{15}	2^{50}
HB	10^{18}	2^{60}
ZB	10^{21}	2^{70}
YB	10^{24}	2^{80}

Page No.

Time

Term	Decimal value	Binary value
Second	1	1
millisecond	10^{-3}	2^{-10}
μ seconds	10^{-6}	2^{-20}
nano seconds	10^{-9}	2^{-30}
pico seconds	10^{-12}	2^{-40}
femto seconds	10^{-15}	2^{-50}

Page No.

1) Suppose we have two implementations of the same computer A has a clock cycle time of 250ps and CPI of 2.0 for some program.
And
Computer B has a clock cycle time of 500ps and a CPI of 1.2 for same program

Which computer is faster and by how much?

Exercise

$CPI_A = 2$ $\quad CPI_B = 1.2$

$T_A = 250 \times 10^{-12}$ $\quad T_B = 500 \times 10^{-12}$

$CT_A = CT_B$

$I_A \times 2 \times 250 \times 10^{-12} = I_B \times 1.2 \times 500 \times 10^{-12}$

$ET_A = 500 \times 10^{-12}$

$ET_B = 600 \times 10^{-12}$ $\quad 1.2 \times 500 \times 10^{-12}$

Comp A is faster 1.2 times

$$MIPS_A = \frac{1}{2 \times 10^6 \times 250 \times 10^{-12}} = \frac{10^6}{500}$$

$$MIPS_B = \frac{1}{1.2 \times 10^6 \times 500 \times 10^{-12}} = \frac{10^6}{600}$$

Page No.

Date ___/___/_____

Saathi

2) A given application written in Java runs in 15 secs on a desktop processor.
A new java compiler is released that requires only 0.6 as many instruction as the old compiler.
Unfortunately it increases the CPI by 1.1
How fast we expect the application to run using this new compiler?

2) $Ex.time_{old}$
$= IC \times CPI \times T = 15 sec$

$Ex.time_{new} = 0.6 \times IC \times 1.1 CPI \times T$

$= 0.6 \times 1.1 \times IC \times CPI \times T$
$= 0.6 \times 1.1 \times 15$
$= 9.9 sec$

5.1 sec faster

$\frac{Ex.time_{old}}{Ex.time_{new}} = \frac{15}{9.9} \approx 1.5151$

Page No.

Date ___/___/_____

Saathi

3) A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:

	Instruction class		
	A	B	C
CPI	1	2	3

For a particular high level language statement, the compiler writer is considering that two code sequences require following instruction counts:

Code seq.	Instruction Mix ($\times 10^9$)		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? — (2)

Which will be faster?

What is the CPI of each sequence?

Page No.

CPI for A = $\frac{(2\times1)+(1\times2)+(2\times3)}{5}$

$= \frac{2+2+6}{5} = \frac{10}{5} = 2$

CPI for B = $(4\times1)+(1\times2)+(1\times3)/6$

$= \frac{4+2+3}{6} = \frac{9}{6} = 1.5$

Ex time A = IC × CPI avg × T

$= \frac{5}{\text{billion}} \times 2 \times T = 10T \times 10^9$

Ex time B = IC × CPI avg × T

$= \frac{6}{\text{billion}} \times 1.5 \times T = 9T \times 10^9$

$\frac{10 \times T \times 10^9}{9 \times T \times 10^9} = 1.111$

4) Suppose we have made the following measurements.

Frequency of FP operations = 25%

Average CPI of FP operations = 4

Average CPI of other operations instructions = 1.33

Frequency of FPSQR = 2%

CPI of FPSQR = 20

Page No.

Assume that two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5. Compare these two design alternatives using processor performance equations.

$\frac{(25\times4)+(2\times20)+(1.33\times75)}{100}$ = Original CPI

$40 + 97.09 = 137.09$

Original $CPI_{avg} = .25\times4 + 0.75\times1.33 = 2$

Design 1 $CPI_{avg} = 2 - 0.02(20-2)$

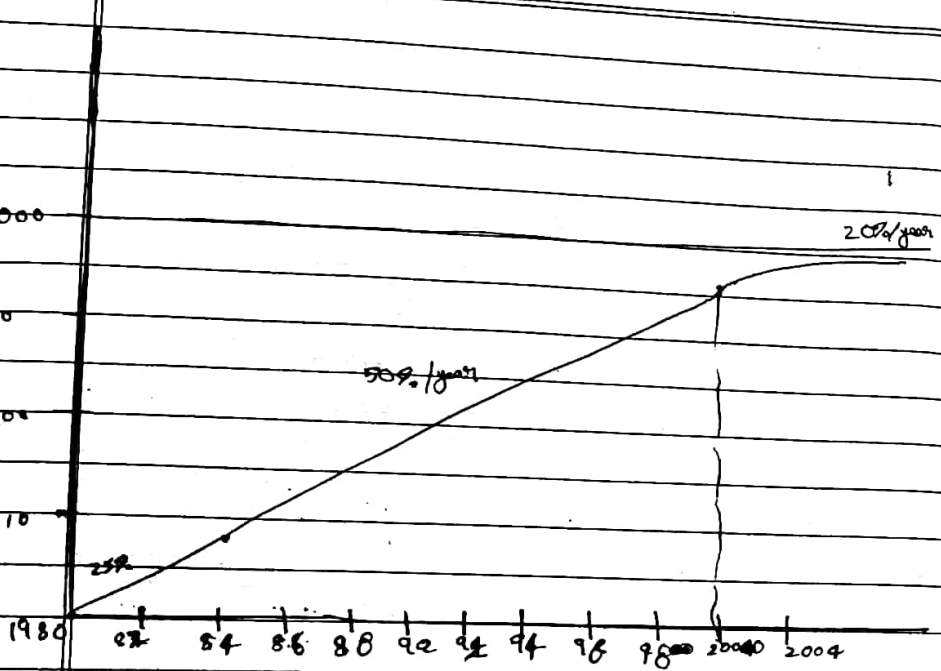
$= 1.64$

Design 2 $CPI_{avg} = 0.25\times2 + 0.75\times1.33 = 1.624$

Design 2 is better

(Common case fast)

Page No.



Trends in Power in Integrated Circuits

$P_{dynamic} = \frac{1}{2} CV^2 f$

C = Capacitative Load

5V → 1V

voltage Scaling

Uniprocessor is now having power problem therefore multiprocessor is introduced.

Date __/__/____

$P_{static} = I_{leakage} \times V$

Summarizing Performance

	Computer-A	Computer-B
Program-1 (sec)	1	10
Program-2 (sec)	1000	100
Arithmetic mean =	$\frac{1+1000}{2}$	$\frac{10+100}{2}$
	500.5 sec	55 sec ✓

	Computer-A	Comp-B	Weightage
Program-1 (sec)	1	10	20%
Program-2 (sec)	1000	100	80%

Weighted arithmetic mean →

Comp (A) (WAM) = $1 \times 0.2 + 1000 \times 0.8 = 800.2$

Comp (B) (WAM) = $10 \times 0.2 + 100 \times 0.8 = 82$ ✓

	Comp-A	Comp-B	Weightage
Prog-1 (sec)	1	10	98%
Prog-2 (sec)	1000	100	2%

Comp-A (WAM) = $1 \times .98 + 1000 \times .02 = 20.98$

Comp-B (WAM) = $10 \times .98 + 100 \times .02 = 11.8$ ✓

$$AM = \frac{1}{n} \sum_{i=1}^{n} Ex\text{-}Time_i$$

$$WAM = \sum_{i=1}^{n} W_i \times Ex\text{-}Time_i$$

System Performance Evaluation Cooperation (SPEC)

Benchmark suite to be run by all producers issued by this cooperation.
This is to check performance of computers.
Include CPU, I/O, etc.

Bench Mark Suite

	Instruction Count (10^9)	CPI
1) Interpreted String Processing	2252	0.6
2) Block Sorting Compression	2390	.376
3) GNU C Compiler	794	2.66
4) Combinational Optimization	221	
5) Go Game (AI)		
6) Search Gene Sequence		
7) Video Compression		
8) XML Parsing		

$$SPEC\text{-}Ratio_i = \frac{T_{ref.\,i}}{T_{Comp\,A-i}}$$ } for each process in Bench Mark Suite

Date ___/___/_____ Saathi

If there are n SPEC-Ratio for a computer,

Geometric mean $= \sqrt[n]{\prod_{i=1}^{n} \text{SPEC-Ratio}_i}$

$\left(\sum = \text{sum}, \ \prod = \text{product}\right)$

$= \sqrt[n]{\text{Ex-Time Ratio}_1 \times \text{Ex-Time Ratio}_2 \ldots \ldots \times \text{Ex-Time Ratio}_n}$

$=$ SPEC Rating

Geometric Mean is used instead of Arithematic Mean because G.M is unbiased to reference comp.

eg;
There are two machines A and B and a reference machine.
There are two Tests T1 and T2 and we obtain the following scores for the machines.

Machines	Test T1	Test T2
Machine A	10 secs	100 secs
Machine B	1 sec	500 sec
Reference Machine	1 sec	100 sec

Page No.

Date ___/___/_____ Saathi

	Test T1	Test T2
Machine A	$\frac{1}{10} = 0.1$	$\frac{100}{100} = 1$
Machine B	$\frac{1}{1} = 1$	$\frac{100}{500} = 0.2$

SPEC Rating$_A$ $= \sqrt[2]{0.1 \times 1} = 0.3162$

SPEC Rating$_B$ $= \sqrt[2]{1 \times 0.2} = 0.4472$

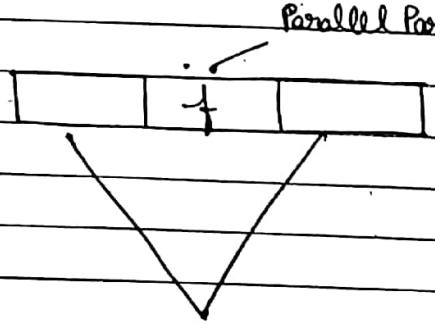
B is better

Performance Improvement By Multiprocessor System

$E_{x\text{-Time}}{}_{\text{Uniprocessor}} = T$

$E_{x\text{-Time}}{}_{\text{Multiprocessor}} \cong \frac{T}{N}$ $(N = \text{No. of Processors})$

Page No.



Original Execution time by Uniprocessor = T_{old}

New Execution time by Multiprocessors

$$= (1-f)T_{old} + \frac{f\,T_{old}}{N}$$

N = No. of processors

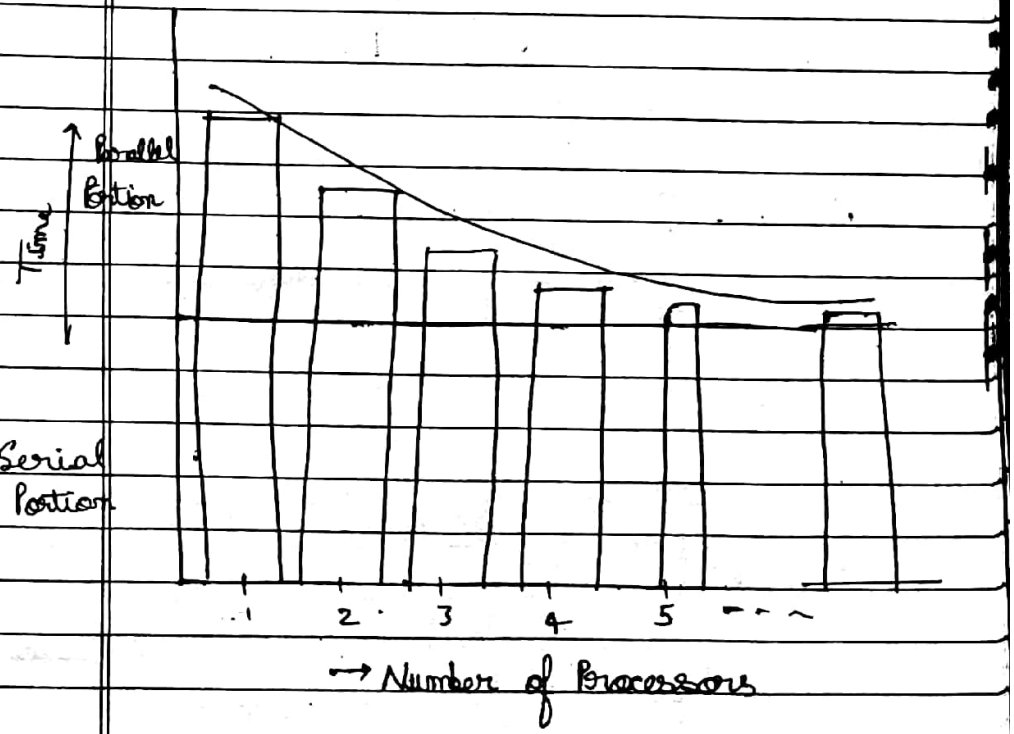
$$\text{Speed Up} = \frac{T_{old}}{T_{new}} = \frac{T_{old}}{(1-f)T_{old} + \frac{f\,T_{old}}{N}}$$

$$\boxed{\text{Speed Up} = \frac{1}{(1-f) + \frac{f}{N}}}$$

Amdahl's Law The amount of speed up is limited by the fraction of the program that can't be parallelized.

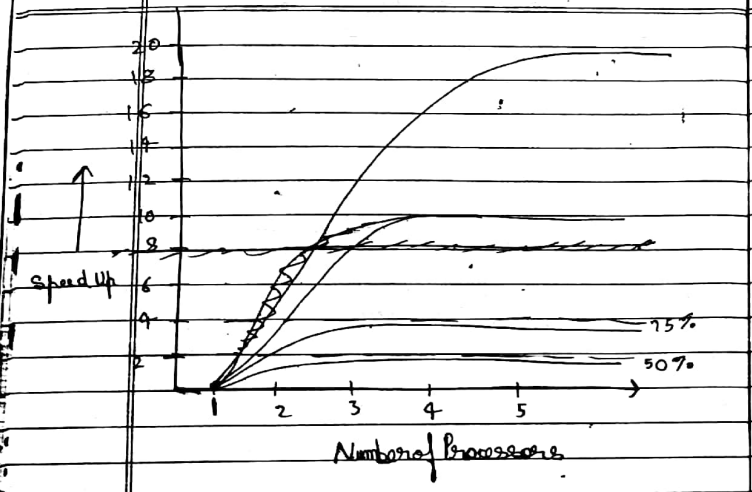
When N is very high ($N = \infty$)

$$\frac{1}{(1-f) + \frac{f}{N}} = \frac{1}{1-f}$$



Saathi

Date ___ / ___ / ____



i) $f = 50\%$ Max speed up $= \frac{1}{1-.5} = 2$ times

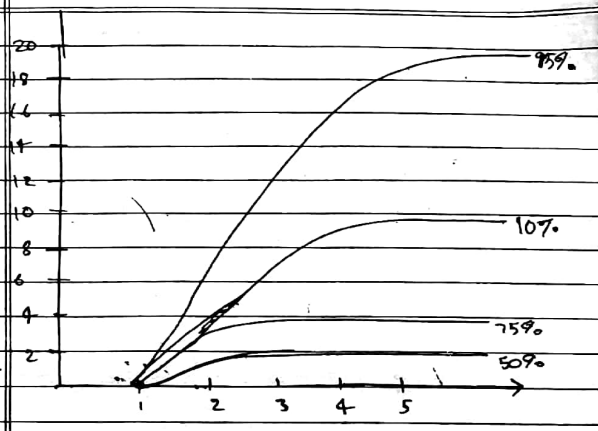
ii) $f = 75\%$ Max speed up $= \frac{1}{1-0.75} = 4$ times

iii) $f = 90\%$ Max speed up $= \frac{1}{1-0.9} = 10$ times

iv) $f = 95\%$ Max speed up $= \frac{1}{1-0.95} = 20$ times

Saathi

Date ___ / ___ / ____



Date __/__/____

Saathi

Amdahl's law

e.g;
Consider an enhancement which runs 20 times faster but which is usable only 15% of the time. What is the overall speed up?

20 times → f = 95%.

15% → 20x

T =

85% of T

15%

$$\text{Speed up} = \frac{1}{(1-0.15) + \frac{0.15}{20}} = 1.166$$

e.g;
Suppose a program spends 80% of its time in a square root routine. How much must you speed up square root to make program 5 times faster?

Page No.

Date __/__/____

$$5 = \frac{1}{(1-0.8) + \frac{0.8}{N}}$$

$$5 = \frac{N}{N(0.2) + 0.8}$$

$$N + 4 = N$$

$$N + 4 = N$$

$$5 = \frac{1}{.2 + \frac{.8}{N}}$$

$$N + 4 = N$$

$$1 + \frac{4}{N} = 1$$

$$\frac{4}{N} = 0$$

$$N = \infty$$

Page No.

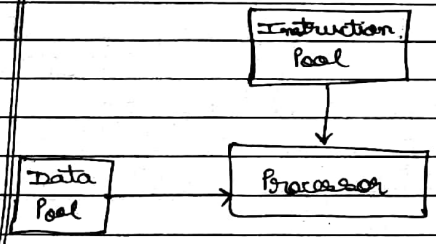
Date ___ / ___ / _____ Saathi

Classification of Multiprocessor Architecture

Flynn's Taxonomy 1968 based on number of instruction stream and data stream.

1) Single Instruction, Single Data (SISD)
2) Single Instruction, Multiple Data (SIMD)
3) Multiple Instruction, Single Data (MISD)
4) Multiple Instruction, Multiple Data (MIMD)

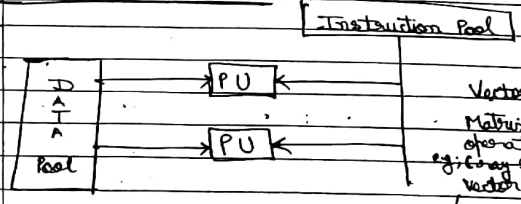
SISD architecture



Uniprocessor

Date ___ / ___ / _____ Saathi

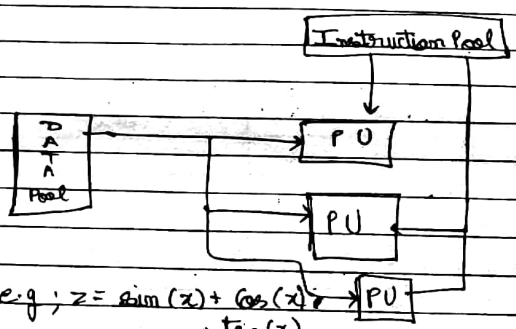
SIMD architecture (Vector Processor)



Multiprocessor

(PU → Processor Unit)

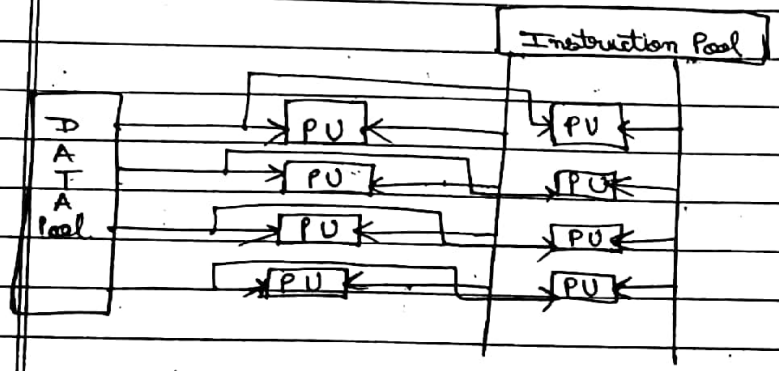
MISD architecture

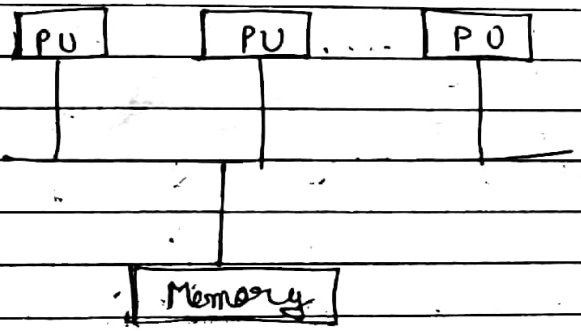


e.g. ; $z = \sin(x) + \cos(x) + \tan(x)$

(very rarely used)

MIMD architecture



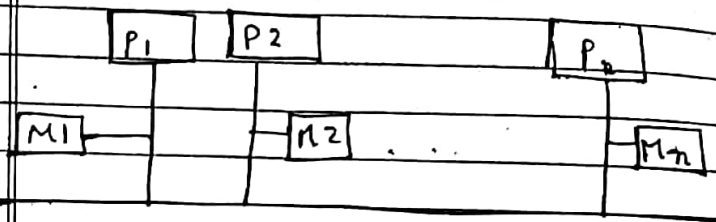


SMP (Shared Memory Processing)
UMA (Uniform Memory Access Architecture)
~~UMA (Uniform Memory Access Architecture)~~

Drawback → for large system congestion is too much

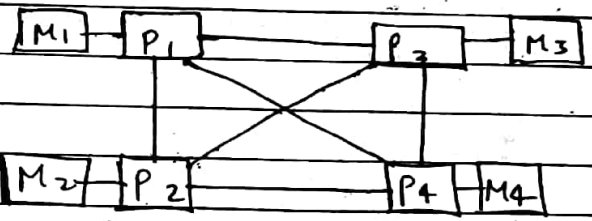
Date __/__/____

NUMA Non Uniform Memory Access Architecture



Only rarely or sometimes access shared memory.

MPI (Message Passing Interface) Architecture



16.8.18 Date __/__/____

MIPS 32 bit ISA (Instruction set Architecture)

ISA → how software & hardware interact

HLL
↓
Compiler (Assembly code)
↓
Assembler
↓
object code (binary)

RISC also called load/store architecture
around 200 different instructions
all are 32 bits long
* Number of Registers = 32
A few data access modes

Design principle → Simple is faster.

MIPS 32 bit design principle

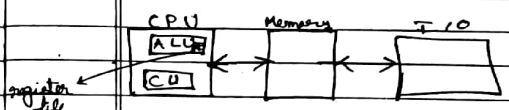
1) Simplicity favours regularity
2) Smaller is faster
3) Make common case fast
4) Good design demands good compromise
(3 different instruction format)

Page No.

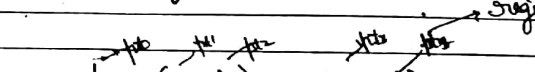
Date __/__/____

CISC — Intel 86 uses this
Very large no of instructions.

We are studying RISC



RF = register file



Compiler assigns register for each variable.

add $t5, $t1, $t4 // $t5 = $t1 + $t4 = i+j

add $t6, $t1, $t2 // $t6 = $t1 + $t2 = g+h

sub $t0, $t6, $t5 // $t0 = $t6 - $t5 f = (g+h) - (i+j)

Load → send data from memory to processor
Store → send data from processor to memory.

Page No.

Date ___/___/____

MIPS Registers and Usage Convention

Name	Number	Use
$zero	$0	$0 = 0 constant value 0
$at	$1	Assembler Temporary
$v0-$v1	$2, $3	values of function results
$a0-$a3	$4-$7	Arguments
$t0-$t7	$8-$15	Temporaries
$s0-$s7	$16-$23	Saved temporaries
$t8-$t9	$24-$25	Temporaries
$k0-$k1	$26-$27	Reserved for OS Kernel
$gp	$28	Global Pointer
$sp	$29	Stack pointer
$fp	$30	Frame pointer
$ra	$31	return address

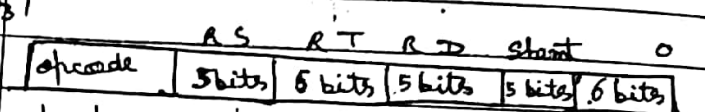
```
int add (int a, int b)
{
    result = a+b;
    return result;
}
```

$a0, $a3 → gets stored in $v0 $v1

Date ___/___/____

Destination, Source, Target

```
add $t5 $t3, $t4
```



Shamt → shift amount
RS → source register
RT → target register
RD → destination register

When 3 registers the instruction format is R format.

MIPS instruction Format

- R Format (3 registers)
- I Format (immediate)
- J Format (Jump) address offset

For R format opcode will have 6 zeroes

Saathi

Date ___/___/_____

opcode	$t3	$t4	$t5	0	function
000000	01011	01100	01101	00000	100000
	$11	$12	$13		add (32)10

add → 32_{10}

Sub $t0, $t6, $t5

↓

$(34)_{10}$

opcode	$t6	$t5	$t0	0	sub
00000	01110	01101	01000	00000	100010

Page No.

Saathi

Date $t0, $t1, $s0, $t2

g = h + A[i], index HLL

A → array

access the memory

$s0 = Addr(A[0]) — already known to compiler

Array (32 bits)

	0	1	2	3	
0	0	1	2	3	A[0]
4	4	5	6	7	
8					
12					A[i]

sll → shift left logic

lw → load word

bytes

Memory is Byte addressable

byte address

sll $t2, $t2, 2 // $t2 = i*4 → shift left 2 times

add $t0, $s0, $t2 — shift left logic

// $t0 = $s0 + $t2 = Addr(A[0] + i*4)

lw $t3, 0($t0)

add $t0, $t3, $t1 // $t0 = $t3 + $t1

g = M(addr(A[0] + i*4)) + h

Page No.

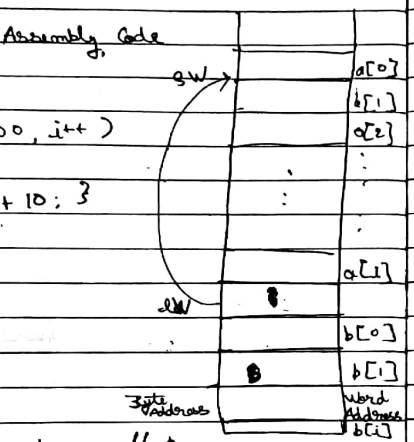
(Tut)

MIPS 32 bit Assembly Code

e.g;

```
for (i=0; i<100; i++)
{
    a[i] = b[i] + 10; }
```

$t0 → i, $s0 → a, $s1 → b




```
addi $t1, $0, 100   // $t1 = 0+100
Loop: sll $t2, $t0, 2   // $t2 = $t0 * 4
                        = i * 4
add $t3, $s1, $t2   // $t3 = $s1 + $t2
                    = addr(B[0]) + i*4
                    = addr(B[i])

lw $t4, 0($t3)      // $t4 = B[i]

addi $t5, $t4, 10   // $t5 = B[i] + 10

addi $t6, $s0, $t2  // $t6 = Addr(A[i])

sw $t5, 0($t6)      // a[i] = b[i] + 10
```

Page No.

```
addi $t2, $t2, 1    // $t2 = $t2 + 1 = i+1
bne $t2, $t1, LOOP
END:
```

e.g;

```
while (i != j)
{
  if (i>j)
    { i = i - j; }
  else
    { j = j - i; }
}
```

$t0 → i, $t1 → j

SLT R1, R2, R3 → if R2 < R3 { R1 = 1 }

~~bne $t1, $t2~~
~~slt $t3, $t1, $t2~~
~~beq~~

```
LOOP: beq $t0, $t1, END
      slt $t2, $t0, $t1   // $t2 = 1 if $t0 < $t1
      bne $t2, $0, ELSE
      sub $t0, $t0, $t1   // $t0 = $t0 - $t1
                          i = i - j
      j LOOP
ELSE: sub $t1, $t1, $t0   // $t1 = $t1 - $t0 = j - i
      j LOOP
END:
```

Page No.

Date ___/___/_____

Saathi

A[i] = A[i] * C[i]

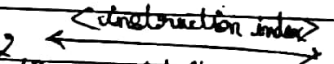
mul

add $to, $t1, $t2

	opcode	RS	RT	RD	Shamt	function
		$t1	$t2	$t0		
>	000000	01001	01010	01000	00000	100000

000000	01010	01100	01110	00000	100010
	$t3	$t4	$t6		Sub

Sub $t6, $t3, $t4



i) add
ii) lw
iii) sll
iv) jump ✓ (∵ only opcode & address)

Date ___/___/_____

Saathi

MIPS 32 Architecture

Type						
R	opcode(6)	rs(6)	rt(5)	rd(5)	shamt(5)	Func(6)
I	opcode(6)	rs(5)	Immediate (16)			
J	opcode(6)	instruction address (26)				

← 32 bit →

rs = source register
rt = target register
rd = destination register
shamt = shift amount

load & store instruction

Mnemonic	format	Encoding
lb	I	32_{10} rs rt ← offset →
lh	I	33_{10} rs rt ← offset →
lw	I	35_{10} rs rt ← offset →
lbu	I	36_{10} rs rt ← offset →
sw	I	43_{10} rs rt ← offset →

Date ___/___/_____ Saathi

MIPS 32 Architecture

Arithematic, logic, instructions

Mnemonic	Format	Encoding
add	R	0_{10} rs rt rd 32_{10}
addv	R	0_{10} rs rt rd 0_{10} 33_{10}
sub	R	0_{10} rs rt rd 0_{10} 34_{10}
and	R	0_{10} rs rt rd 0_{10} 36_{10}
or	R	0_{10} rs rt rd 0_{10} 37_{10}
XOR	R	0_{10} rs rt rd 0_{10} 38_{10}
addi	I	8_{10} rs rt ← offset →

e.g.; 101010
101001
101000

andi	I	12_{10} rs rt ← offset →

Shift instruction

Mnemonic	Format	Encoding
SLL	R	0_{10} 0_{10} rt rd Sa 0_{10}
SRL	R	0_{10} 0_{10} rt rd Sa 2_{10}
SRA	R	0_{10} 0_{10} rt rd Sa 3_{10}

Page No.

Date ___/___/_____ Saathi

Jump & Branch

Mnemonic	Format	Encoding
J	J	2_{10} ← instruction index →
JAL	J	3_{10} ← instruction index →
beq	I	4_{10} rs rt ← offset →
bne	I	5_{10} rs rt ← offset →

Page No.

Date ___/___/_____ Saathi

28.8.18

e.g:

```
if (i == j)
{
   f = g + h;
}
else
{
   f = g - h;
}
```

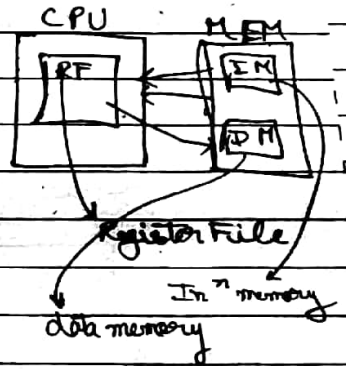
```
bne $t0, $t1, ELSE
add $t2, $t3, $t4
 J END
ELSE: Sub $t2, $t5, $t4
END
```


Date ___/___/_____ Saathi

MIPS 5 stage Pipeline

add R1, R2, R3 // R1 = R2 + R3

1) Instruction Fetch
2) Instruction Decode => add R2, R3
3) Instruction Execute => R2 + R3
4) NOP (No operation)
5) Write Back => R1 = R2 + R3



lw R1, 1032(R3)

1) Instn Fetch
2) ID => R3
3) Ex => 1032 + R3 (address calculation)
4) Memory Access
5) WriteBack R1 = value [1032(R3)]

SW R1, 32(R3)

1) IF
2) ID
3) Ex => 32+R3 (address)
4) Mem => Value (adder(32+R3)) ← R1
5) NOP

beq R1, R3, LOOP → PC related (Program Counter)

1) IF
2) ID
3) Ex => R1 is compared with R3
4) MA P
5) NO P

add R1, R2, R3	F D E M W
sub R4, R5, R6	F D E M W
and R7, R8, R9	F D E M W
XOR R11, R12, R13	F D E M W

8 cycles for 4 instructions

CPI = 2

Page No.

Date __/__/____

Saath

Throughput = IPC = $1 \frac{4}{8} = \frac{1}{2}$

Non pipeline (IPC) = $\frac{1}{5}$

Pipeline is better.

Speed up = $\frac{T_{\text{Non Pipeline}}}{T_{\text{pipeline}}} = \frac{20}{8} = 2.5$

Maximum speed up is 5 times.
↑ stages ↑ speed up

Pipeline Hazards → when maximum benefit from pipeline can't be obtained.

1) Structural Hazard → (2 process when requiring same part)

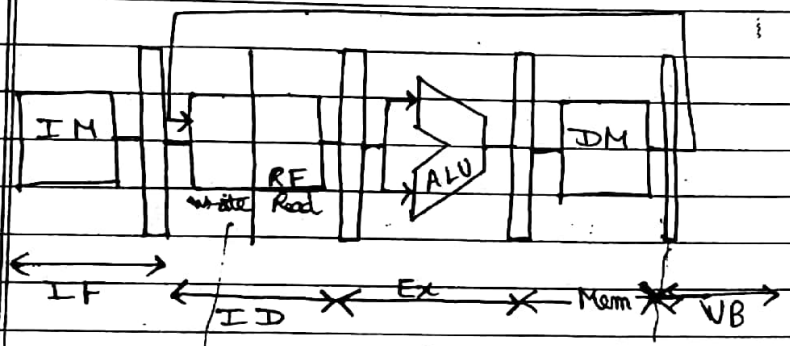
2) Data Dependency →

3) Controlled Hazard → Branch,

Date ___/___/_____

Saathi

MIPS 5-stage Pipeline



Read & write both require it
In half cycle write & in another half read takes place

Harvard architecture followed

By modifying hardware structural hazard is removed.

Date ___/___/_____

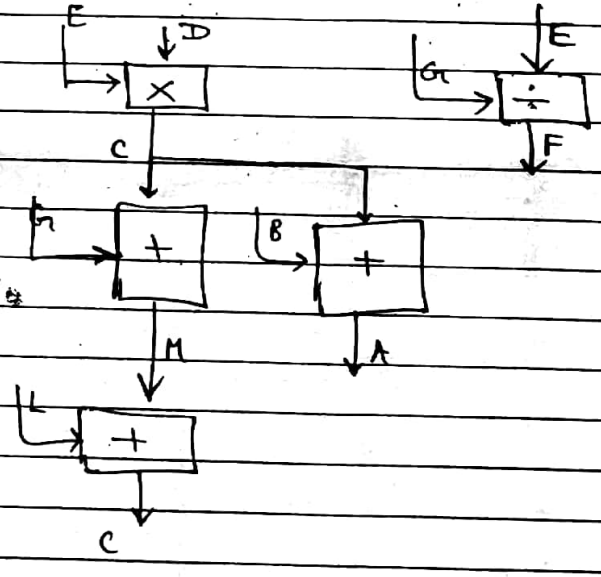
Saathi

Data Dependency

S1 : C = D × E
S2 : M = G + C
S3 : A = B + C
S4 : C = L + M
S5 : F = G ÷ E

S1 & S2 } not ||el ∴ can't work same time
S3 } parallel

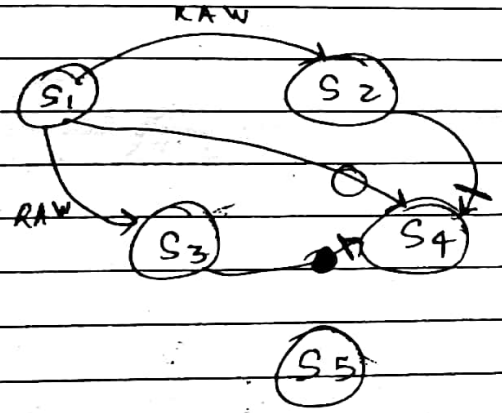
Data Flow Diagram



Types of Data Dependency

1) Read after Write (RAW) dependency. ⟶
2) Write after Write (WAW) dependency. (output dependency) ⟶O⟶
3) (S4, S3) / (S3, S4) Write after Read (antidependency) ⟶|⟶

Dependency graph



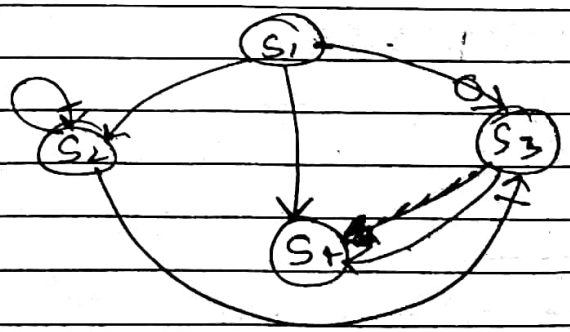
Bernstein's Condition for Parallelism

I_1, I_2	$I_1 \cap O_2 = \phi$
O_1, O_2	$I_2 \cap O_1 = \phi$
	$O_1 \cap O_2 = \phi$

I → Input
O → Output

C = D × E
↓ (D × E)
O I

S1: load R1, A // R1 = A
S2: add R2, R1 // R2 = R1 + R2
S3: Move R1, R3 // R1 = R3
S4: store B, R1 // B = R1

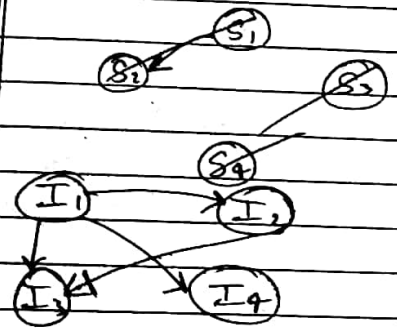


29.8.18 Date __/__/____

Saathi

Program

I1: lw r1, 0(r2)
I2: sub r4, r1, r6
I3: and r6, r1, r7
I4: or r8, r1, r9



F D E M W
F - - - D

16 Cycles
Inst → 4
4 C P I

but we do Forwarding

F D E M W
F - D E M W
F D E M W
F D E M W

CPI = 9/4 = 2.25

IPC = 4/9 = 0.44

Page No.

Date __/__/____

Saathi

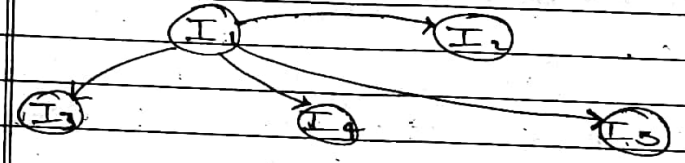
Stalling is done to cope with hazard

Forwarding is mechanism so that it can be used when available & reduce stalls. (Load Use Hazard)

Program

I1 add r1, r2, r3
I2 sub r4, r1, r3
I3 and r6, r1, r7
I4 or r8, r1, r9
I5 XOR r10, r1, r11

F D E M W
F - D E M W
F D E M W
F D E M W
F D E M W



F D E M W
F - D E M W
F F D E M W
F D F M W
F D E F M W

F D E M W
F F E M W
F D F M W

Page No.

Date __/__/____

```
F D E M W
  F D E M W
    F D E M W
      F D E M W
        F D E M W
```

Cycles = 9
Inst^{n} = 5

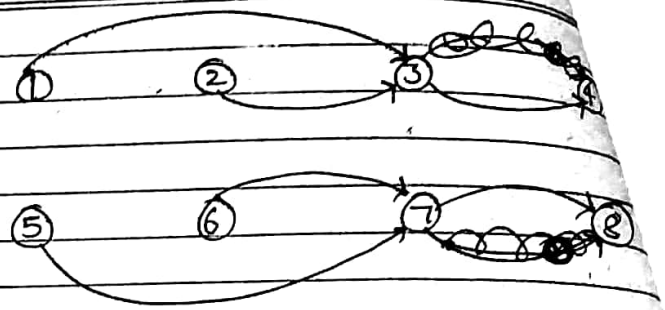
$$CPI = \frac{9}{5}$$

$$IPC = \frac{5}{9}$$

Program

1)	lw $t1, 0($s0)	F D E M W
2)	lw $t2, 0($s1)	F D E M W
3)	add $t3, $t1, $t2	F - D E M W
4)	sw $t3, 0($s2)	F D E M W
5)	lw $t4, 0($s3)	F D E M W
6)	lw $t5, 0($s4)	F D E M W
7)	sub $t6, $t4, $t5	F - D E M W
8)	sw $t6, 0($t5)	F D E M W

Page No.



Forwarding is done / can be done from M only

$$CPI = \frac{8}{8}$$

$$IPC = \frac{8}{14}$$

Page No.

Saathi

Date ___ /___ /_____

Code reordering is done to cope up with load-use hazard.

Reordered code

```
lw $t1, 0($s0)      F D E M W
lw $t2, 0($s1)      F D E M W
lw $t4, 0($s3)        F D E M W
lw $t5, 0($s4)          F D E M W
add $t3, $t1, $t2         F D E M W
sw $t3, 0($s2)              F D E M W
sub $t6, $t4, $t5             F D E M W
sw $t6, 0($s5)                  F D E M W
```

$CPI = \frac{12}{8}$

$IPC = \frac{8}{12}$

Page No.

Saathi

Date ___ /___ /_____

Hazard control techniques:

1) Forwarding
   (Load-Use Hazard)
2) Code reordering
   (Stall in branch - (Branch Hazard))
3) Branch Prediction
   - Branch Taken
   - Branch Not Taken
4) Delay Slot
   (Branch Hazard solution)
5) Loop unroll

Branch Hazard
1) Branch prediction
2) Delay slot
3) Loop unrolling

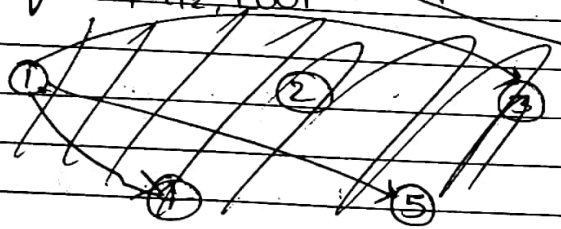
Page No.

Program

1) beq $r_1, r_3,$ [redacted] 36
2) and r_2, r_3, r_5
3) or r_6, r_1, r_7
4) add r_8, r_1, r_9
5) XOR r_{10}, r_1, r_{11}
6) [redacted] beq r_{11}, r_{12}, LOOP

comparison & add(36) both at D

F D E M W
- F D E M W



if not dependent then can execute 6 before 5 to prevent stall i.e; delay

No dependency ∴ beq do not write r_1 > r_3

it has to stall

$t0

for(i=1000; i>0; i=i-1)
{
x[i] = x[i] + s }

$s0 ↑ $s1

addi $t1, $0, $1000
sll $t2, $t0, 2
add $t3, $s1, $t2
lw $t4, 0($t3)

LOOP: add $t1, $0, $0 // $t1 = 0
sll $t2, $t0, 2 // $t2 = $t0 * 4 = i * 4
add $t3, $s0, $t2 // $t3 = Addr(x[0]) + i*4 = Addr(x[i])
lw $t4, 0($t3) // $t4 = x[i]

(load use hazard)

addd $t5, $t4, $s1 // x[i] = x[i] + 5

subi $
addi

sw $t5, 0($t3) // x[i] = x[i] + 5

subi $t0, $t0, 1 // t0 = t0 - 1
t0 = i - 1

can be reordered

Page No.

bne $t0, $0, LOOP
END

Branch always stall for one.

Cycles

5 + (12 X 1000) + 1000
= 5 + 12000 + 1000
= 13005

Loop unrolling

```
for(i=1000 ; i>0 ; i=i-4)
{
    x[i] = x[i] + s;
    x[i-1] = x[i-1] + s;
    x[i-2] = x[i-3] + s;
    x[i-3] = x[i-3] + s;
}
```

```
add $t1, $0, $0     // t1 = 0
sll $t2, $t0, 2     // t2 = i*4
add $t3, $s0, $t2   // t3 = addr(x[i])
lw $t4, 0($t3)      // t4 = x[i]
lw $t5, -4($t3)     // t5 = x[i-1]
lw $t6, -8($t3)     // t6 = x[i-2]
lw $t7, -12($t3)    // t7 = x[i-3]
add $t8, $t4, $s1   // t8 = x[i]+s
add $t9, $t5, $s1   // t9 = x[i-1]+s
add $s2, $t6, $s1   // s2 = x[i-2]+s
sw $t8, 0($t3)      // x[i] = x[i]+s
add $s3, $t7, $s1   // s3 = x[i-3]+s
sw $t9, -4($t3)     // x[i-1] = x[i-1]+s
sw $s2, -8($t3)
sw $s3, -12($t3)
subi $t0, $t0, -4   // i = i-4
bne $t0, $0, LOOP
```

Scanned by CamScanner

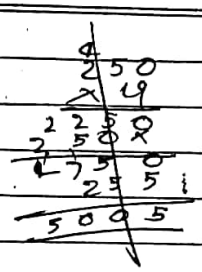
Date __/__/____

Saathi

No of cycles

$5+(49\times 50)+250$

$= 5005$



Pipeline is overlapping it is not complete parallelism.

Date __/__/____

Saathi

Super Scalar Architecture / Super Scalar Processor

I_1: F D E M W
I_2: F D E M W
I_3: F D E M W
I_4: F D E M W
I_5: F D E M W
I_6: F D E M W

CPI: $\frac{7}{6}$

IPC: $\frac{6}{7}$ (Compilor based)

Multiple register ports, multiple fetchor, decodor etc.

Very large Instruction Word (VLIW) Processor

I_1	I_2
ALU or Branch Instruction	LW/SW instruction

Saathi

Date __/__/____

	Slot 1	Slot 2
1)	add $t1, $0, $0	
2)	sll $t2, $t0, 2	
3)	add $t3, $S0, $t2	lw $t4, 0($t3) (dependency)
4)	bne $t0, $0, -4 (addi $t0, $0, -4)	lw $t4, 0($t3)
5)	add $t6, $t4,	lw $t5, -4($t3)
6)	add $t8, $t4, $S1	lw $t6, -8($t3)
7)	add $t9, $t5, $S1	lw $t7, -12($t3)
8)	add $S2, $t6, $S1	sw $t8, 0($t3)
9)	add $S3, $t7, $S1	sw $t9, -4($t3)
10)	bne $t0, $0, LOOP	sw $S2, -8($t3)
11)		sw $S3, -12($t3)

Cycles
= 5 + (10 × 250)
= 2505

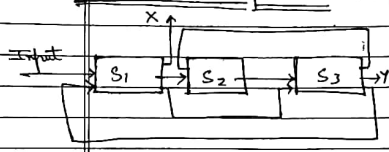
Page No.

4.16.18

Date __/__/____

Saathi

Non-Linear Pipeline



feedback paths

1) Streamline path
2) Feedback path
3) Feedforward path
4) Multiple outputs from any stages of pipeline

Linear Pipeline

	1	2	3	4	5	6
F	I_1	I_2				
D		I_1	I_2			
E			I_1	I_2		
M				I_1	I_2	
W					I_1	I_2

Reservation Table for function X

	1	2	3	4	5	6	7	8	9
S_1	X	X				X		X	
S_2		X		X					
S_3			X	X	X		X		

Page No.

Reservation table for function Y

	1	2	3	4	5	6	7
S_1	Y				Y		
S_2			Y				
S_3		Y		Y		Y	

	1	2	3	4	5
S_1	X	2	3		
S_2		X		X	
S_3			X	2	3

collision

not allowed

2 ins^n can't be assigned at same time.

Forbidden latency for function X

Stage 1: 5 is forbidden latency.

5, 7 are forbidden latency.

5 7 2

Reservation Table for X

	1	2	3	4	5	6	7	8	9
S_1	X					X		X	
S_2		X		X					
S_3			X		X		X		

Stage 1: 5 7 2

(6-1) (8-1) (6-6)

Stage 2: 2

(4-2)

Stage 3: 2 4

(5-3) (7-3)

Stage 4: 2 4 5 7

all forbidden latency

Collision Vector

C_7 C_6 C_5 C_4 C_3 C_2 C_1

1 0 1 1 0 1 0

State diagram

1 0 11 0 1 0

Right shift

0 will come out, if 0 comes out then no collision you can send a new instruction

1 0 1 1 0 1 0

0 1 0 1 1 0 1 ← on bit: right shift

OR 1 1 1 1 1 1 1

all

1 0 1 1 0 1 0

0 bit right shift

0 1 1 0 1 0

0 0 0 1 0 1 1

1 0 1 1 0 1 1

1 bit right shift

1 0 1 1 0 1 0

0 1 0 1 1 0 1

OR 1 1 1 1 1 1 1

	1	2	3	4	5	6	7	8	9
S_1	X			X		X		X	
S_2		X		X	X				
S_3			X		X	X	X	X	

	1	2	3	4	5	6	7	8
S_1	X	X				X		X
S_2		X	X	X	X			
S_3			X	X	X	X	X	

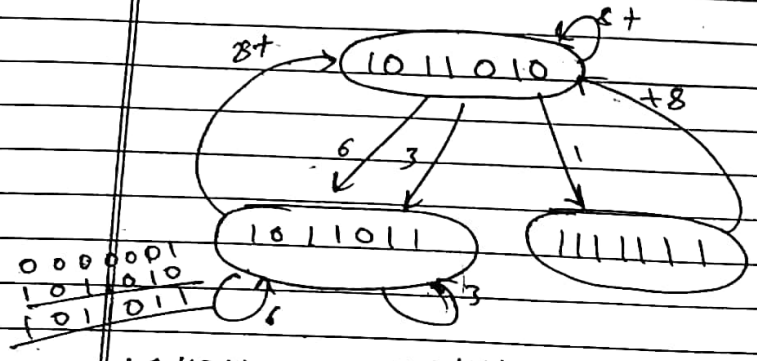
6 bit right shift

1 0 1 1 0 1 0

0 0 0 0 0 0 1

1 0 1 1 0 1 1

Scanned by CamScanner



Simple Cycles: (3,8) (6,8) (1,8) (8)
(3,3) (6,3) (6,6)
(3,6)

	Cycles
(3,8) →	11
(6,8) →	14
(1,8) →	9
(3,3) →	6
(6,3) →	9
(6,6) →	12
(3,6) →	9

Minimum average latency (MAL)

$= \frac{3+3}{2} = 3$

after every 3 cycle a new instruction can be issued with no collision.

Greedy Cycle → no other option

(3,8) , (6,8) , (1,8) , (3,3)

(6,3) (3,6) (6,6)

	1	2	3	4	5	6	7	8	9
S_1	X			0		X		X	0
S_2		X		X	0		0		
S_3			X		X	0	X	0	

Nonlinear Pipeline Scheduling

a) What is throughput of this pipeline.?

Reservation Table

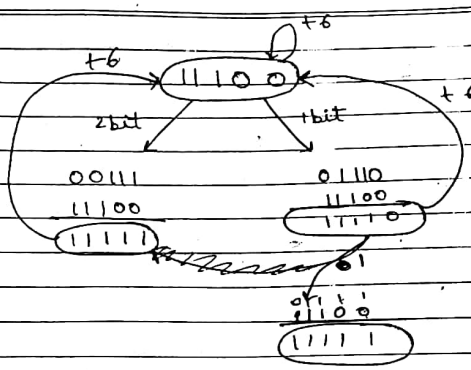
	1	2	3	4	5	6	
S_1	X	0	1			X	
S_2		X	0	1	X	0	2
S_3			X	0	1		
S_4				X	0	1	
S_5		X	0	2		X	2 Colliding

S_1: $6-1$
$= 5$
S_2: $5-2$
$= 3$
S_3:
S_4:
S_5: $6-2$
$= 4$

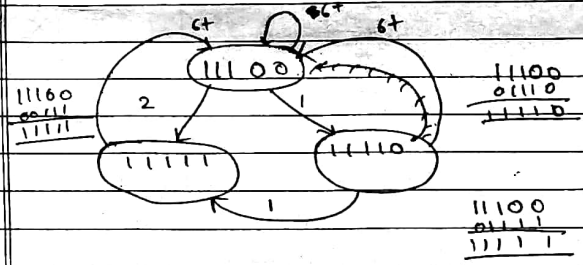
for X → first instⁿ

Forbidden = {3, 4, 5}
latency

C_5 C_4 C_3 C_2 C_1
1 1 1 0 0 → Collision vector



01111
11100
11111



Simple cycles: (2,6), (1,6), (6), (1,1,6)

Greedy cycle: (2,6), (1,6), (6), (1,1,6)

Minimum average latency:
(2,6) → 8/2 = 4 (1,1,6) → 8/3 = 3
(1,6) → 7/2 = 4
(6) → 6 = 6

Date ___/___/____

MAL = 3

Throughput = No of cycles per instruction

Per 3 Cycles → 1 instruction is executed

$= \frac{1}{3}$

Q)

	1	2	3	4	5	6
S_1	X			X		X
S_2		X			X	
S_3	X		X			
S_4				X		X
S_5		X				

S_1 : 3 (4-1), 5 (6-1), 2 (6-4)

S_2 : 3 (5-2)

S_3 : 2 (3-1)

S_4 : 2 (6-4)

S_5 : None

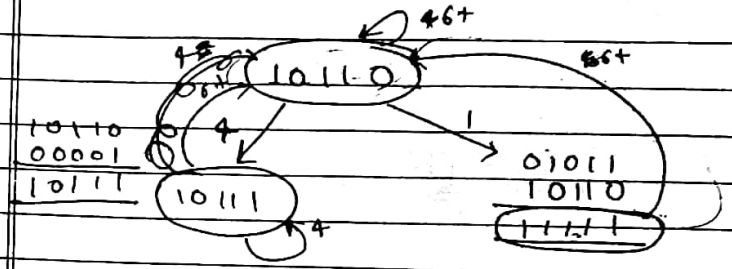
Page No.

Date ___/___/____

{2, 1, 3, 5}

$C_5\ C_4\ C_3\ C_2\ C_1$

1 0 1 1 0



10111
00001
10110

Simple cycles: {6}, {1,6}, {4,4}, {4,6}, {4,4,6}

Greedy cycles: {6}, {1,6}, {4,4}, {4,6}, {4,4,6}

Average L

6 → 6

{1,6} → 7/2 ≃ 4

{4,4} → 8/2 ≃ 4

{4,6} → 10/2 = 5

{4,4,6} → 14/3 ≃ 5

MAL : 4

Throughput = 1/4

Page No.

Scanned by CamScanner

Saathi

Date ___/___/_____

		1	2	3	4	5	6
(a)	S_1	Y				Y	
	S_2			Y			
	S_3		Y		Y		Y

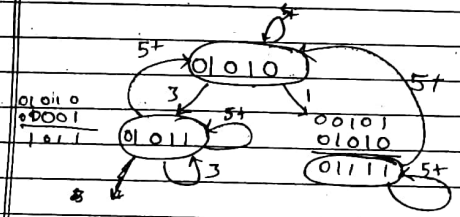
S_1: (5-1) = 4

S_2: None

S_3: (4-2) = 2
(6-4) = 2
(6-2) = 4

{2, 4}

C_5 C_4 C_3 C_2 C_1
0 1 0 1 0



01011
00001
01011

Simple cycles: {5} {1,5} {3,3} {3,3,5} {3,5}, {1,3,5}, ~~{3,3,3}~~

~~Greedy cycles: all simple cycles here~~

Page No. ☐

Saathi

Date ___/___/_____

AL:

5 → 5

{1,5} → $\frac{6}{2} = 3$

{3,3} → $\frac{6}{2} = 3$

{3,5} → $\frac{8}{2} = 4$

{3,3,5} → $\frac{11}{3} \simeq 4$

Throughput = $\frac{1}{3}$

Page No. ☐