L-1/19.07.2023

L-02/24.07.2023

Address: Unique codes that represent a memory location.

RAM

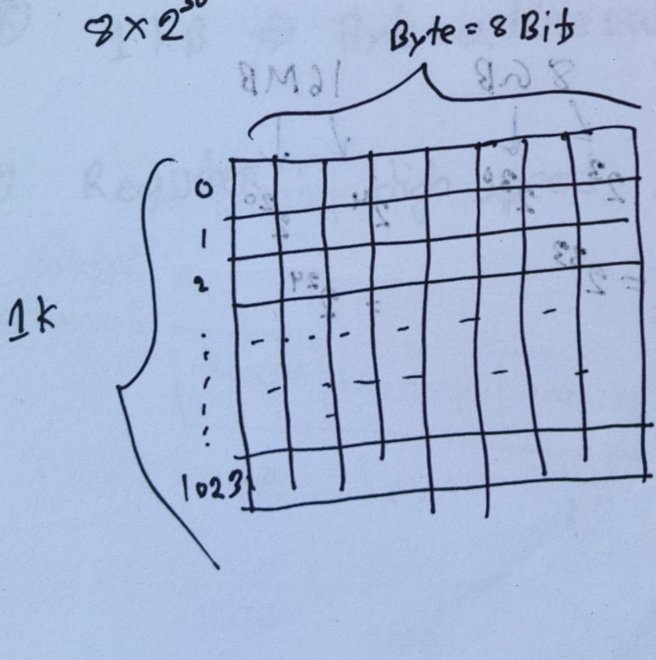
8 GB

Number of Addressable location

8×2^{30}

Content of each location

1 Byte = 8 bits



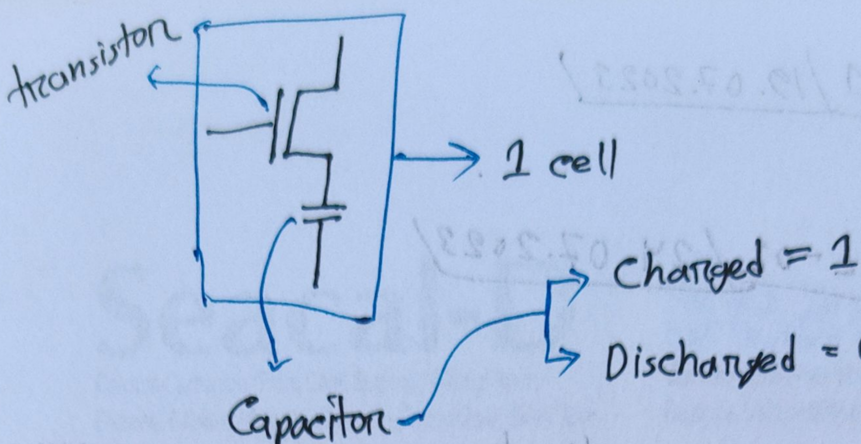
Memory ⇒ 1 kB

⇒ 1k = Row = 1024

B = 8 bits = 8 collumn

Cell = 1024 × 8

= 8192



⊛ 1 kB

How many bits we need to address all addressable location?

$\Rightarrow$ $2^{10} = 1023 = \boxed{1111111111} \rightarrow$ 10 bits

address $\Rightarrow$ 0000000000 – 1111111111

⊛

1 MB → 2^0, 2^{20} $= 2^{20}$

1 GB → 2^0, 2^{30} $= 2^{30}$

8 GB → 2^3, 2^{30} $= 2^{33}$

16 MB → 2^4, 2^{20} $= 2^{24}$

8 kB → 2^3, 2^{10} $= 2^{13}$

⇒ Then we needs 13 bits to represent all address

⇒ if we used 12 bits then we can access half of the memory.

⇒ every bits will divide the memory by 2.

⊛ Byte addressable memory?

⇒ if content of each addressable location is 1 Byte or 8 bits.

Ex.: 16 MB

⊛ 1k × 16 bits ⇒ Word addressable

⊛ 1 kB ⇒ Byte addressable

⊛ Register : High speed storage inside the CPU.

L-03 / 26.07.2023 /

⊛ BUS System : 8 bit / 16 / 32 bits

Deta BUS : Transfer or carry Deta

Address BUS : 10 bits ⇒ it can handle $2^{10} = 1024$

= 1 kB memory

⇒ Decide maximum memory limit.

⊛ Important Topics :

⊛ Registers

⊛ Memory (RAM)

⊛ Memory Address

⊛ Instruction ⇒ Comand for basic operation

Construction

Store Location

	Operation	Operand	Operand	Result
Low level ⇒	ADD	R1	R2	R3
Binary Code ⇒	0010	0001	0010	0011

⊛ Deta ⇒ Operands used in operation

⊛ Machine Code ⇒ Size depends on CPU.

⇒ minimum 8 bit

⊛ Size of Machine Code: Using Byte addressable memory

⇒ If machine code of an instruction is represent in 8 bit, then we need 1 location in memory.

⇒ 16 bits ⇒ 2 location

⇒ 32 bits ⇒ 4 location

⊛ Most frequent operation ⇒ Read/Write

⊛ Accumulation Register ⇒ Only for Read/Write
Pre-fixed.

⊛ General Register ⇒ any auto generated.

LOAD = read from RAM

Store ⇒ write in RAM

⊛⊛⊛

sum = 5 + 3 = 8

High Level Code:

sum = a + b ;

only one instruction

sum, a, b are variable

Stored in RAM

Address :

$\left.\begin{array}{l} \text{sum} = 12 \\ a = 10 \\ b = 11 \end{array}\right\}$ Assume that

Low Level Code:

LOAD R1, 10

LOAD R2, 11

ADD R1, R2

STORE 12, R3

Comand destination, Source

R1, R2, R3 are Registers

became four instruction

Machine Code:

Address			
100	00101	0001	1010
101	00101	0010	1011
110	01011	0001	0010
111	00111	1100	0011

Stored in memory Sequentially

Assume that

LOAD = 00101

STORE = 00111

ADD = 01011

R1 = 0001

R2 = 0010

R3 = 0011

Execution:

Picture from slide page 21

1. Address ~~from th~~ of the 1st instruction is loaded in program center. (100)

2. Then the address placed in address BUS (100)

3. Then control unit generate a signal to read in control BUS.

4. Then the instruction from RAM read and placed in Deta BUS (1st instruction)

5. Then it comes to internal BUS. (registers)

6. After that instruction send to the instruction Register.

7. Instruction Register decode the instruction and control unit generate some ~~siglat~~ signal sequentially.

8. Then control unit get the address and placed it in the address Bus (10) thru the internal Bus.

9. Then send a signal to read thru the control Bus. (Load)

10. Then the deta is copied and placed in the Deta Bus (0101 → 5)

11. Then the deta send to the Register number 1.

12. Same process to Load the b = 11 = content 3

13. Then a signal generate to Add them. Then the deta again assigned in the registers of ALU thru the internal Bus.

14. Then Add signal send to the ALU and ALU add them and placed in the output Register.

15. ~~A~~ Now its start to store process.

16. 1st controll unit send the address in the RAM.

17. Once location selected, it ~~co~~ placed the deta in ~~data~~ deta Bus. ~~Then~~

18. Then send a signal to write in the RAM.

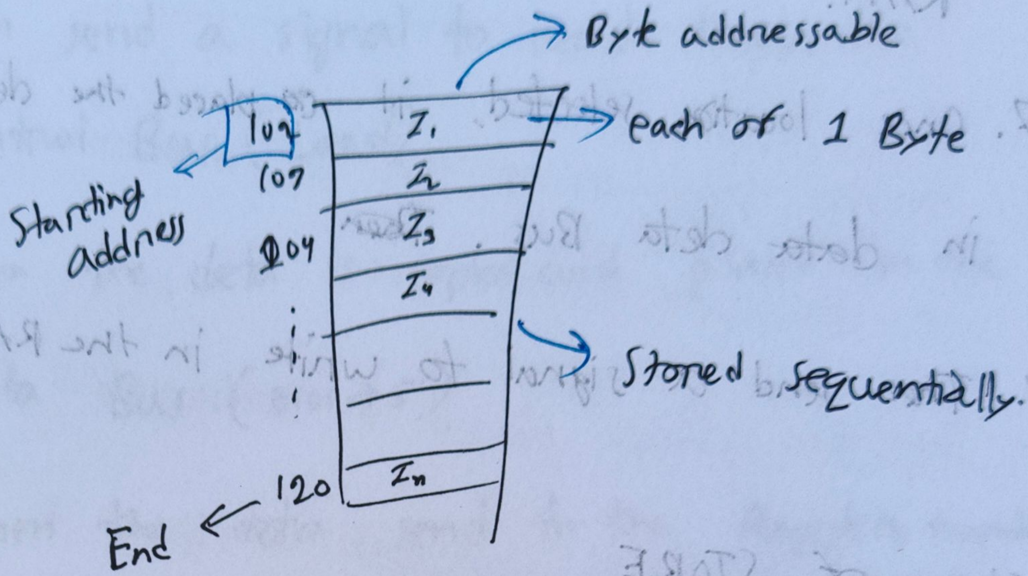
Steps of STORE

1. Decode instruction
2. Address placed to address Bus (05)
3. Deta placed to Deta Bus R2
4. CPU will ~~generta~~ activate write signal.
5. Deta copied from the deta Bus to RAM at address 05.

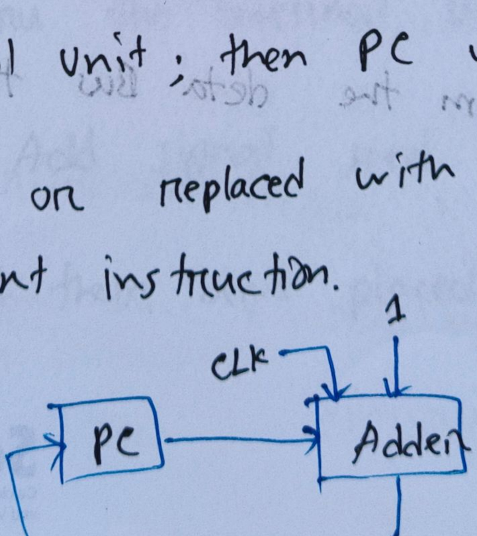
⊛ Program Counter:

a list of sequential instruction

Whenever a program is loaded into RAM



1. Loaded in PC
2. Address Bus
3. Controll Bus (Read)
4. Instruction loaded in IR
5. control unit; then PC will increament by 1 or replaced with the ~~next~~ address or next instruction.



PC always ~~to~~ hold **Address** *** or next instruction.

It is automatically increament for the next instruction are set by the low level code. increament depends on command size. If it takes ~~to~~ two location to store a instruction, then PC will increament by 2.

Test on Next Class

L-04 / 31.07.2023 /

⊛ How does the CPU run a program?

⊛ How does the CPU process an instruction?

⊛ What is instruction Cycle? Explain.

⊛ Must for next class Test.

Program (Low level language)

↳ a list of instruction

↳ command for a basic operation.

↳ arithmatic, logical

Operation	Operand 1 or Address	Operand 2 or Address	result or address

Address ⇒ Name of Register or memory location.

$$\left.\begin{array}{l} \text{ADD } R_1\ R_2\ R_3 \\ \text{ADD } 5, 7 \\ \text{ADD } M_1, M_2, M_3 \end{array}\right\}$$ Sample command of three type.

Operation → Sample Code for Operation

- ADD → 0001
- STORE →
- READ →
- MUL →
- DIV →

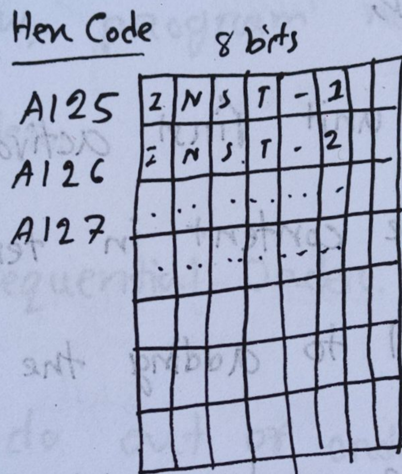
⊛ everything have a code identity.

These are machine code.

Program

- Inst-1 → 1 Byte
- Inst-2 → 8 bits
- ⋮
- Inst-n →

RAM - Byte addressable



⊛ The program must be loaded into RAM with a specific starting address sequentially.

⊛ Program counter is loaded with the address of 1st instruction of a program.

⊛ Program counter always hold $\boxed{\text{address}}$ of program instruction.

⊛ The CPU will read the machine code of 1st instruction from the specific address of RAM.

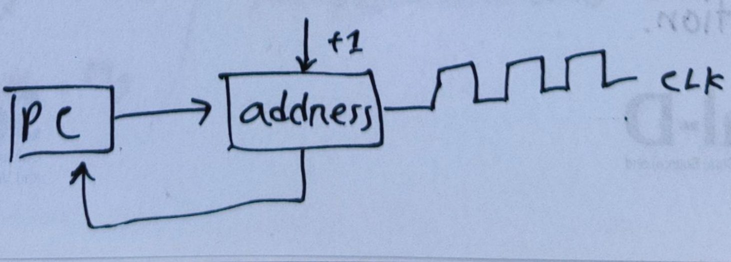
⊛ Machine code is loaded into instruction Register. Then the Register will decode the instruction in Control unit. And the controll unit will generate a sequence of signal according to the instruction.

For ADD,

controll unit first activate the adder in ALU then set the content in temporarily Register. Then send a signal to adding the content and ALU will do it. Program center will increament by 1 to point the next instruction.

PC ⇒ Register inside the CPU

After read a instruction PC will increament or replace by the address of next instruction.



VNB

1. Program and Data converted into Binary and stored in a single memory.

2. Single Bus System.

3. Sequential

* It is important to know, program need to load in sequential order.

VNB → strictly follow sequential Order.

* Modern Processor can do out of order process.

* $\boxed{\text{Diagram}}$ need to explain Program process, VNB etc.

Harvard Architecture:

Seperate memory for program and data.

Double Bus System.

$\boxed{\text{Harvard Vs VNB}}$

L-5/02.08.2023/

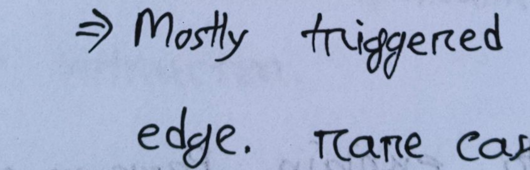
⊛ For a processor 1 second time is huge - can do millions operation.

⊛ 1 sec

$$\left.\begin{aligned} 10^{-6} &= \text{micro} \\ 10^{-9} &= \text{nano} \\ 10^{-12} &= \text{pico} \end{aligned}\right\} \text{used to represent processing time.}$$

⊛ CPU Clock ⇒ Highly precise square wave ~~sigla~~ signal.

⇒ Mostly triggered in rising or droping edge. rare case of level ~~edge~~ triggered.

⊛ 

Full cycle of signal or 1 cycle

T = time needs to complete a cycle

= Time period

$= 0.2 \text{ sec} = \frac{1}{f}$

f = frequency (Hz)

= ~~ho~~ how many cycle is completing in 1 sec.

$= \frac{1}{T} = \frac{1}{0.2} = 5 \text{ Hz}$

1000 Hz = 1 kHz

$10^6 = 1 \text{ MHz}$

$10^9 = 1 \text{ GHz}$ → frequently used in modern processor

$10^{12} = 1 \text{ THz}$

Clock frequency, f = 2 MHz

$\therefore$ Time period, $T = \frac{1}{f} = \frac{1}{2 \times 10^6}$ sec

$= 0.5 \times 10^{-6}$ sec

$= 0.5 \ \mu\text{sec}$

* Time period, $T = 0.33\ \mu sec$

$$\therefore f = \frac{1}{T} = \frac{1}{0.33 \times 10^{-6}} = 3.33 \times 10^{6}\ Hz$$
$$= 3.33\ MHz$$

* $f = 2\ MHz$

single additional operation takes $2\ \mu sec$.

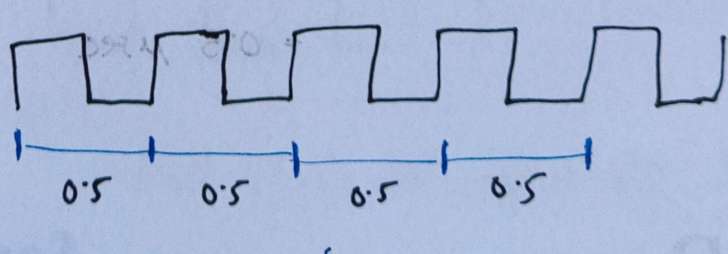
How many CPU Clock cycle are required to perform an additional operation?

$\Rightarrow$ $$T = \frac{1}{f} = \frac{1}{2 \times 10^{6}} = 0.5 \times 10^{-6}\ sec$$
$$= 0.5\ \mu sec$$

$\therefore$ cpu clock required $= \frac{2}{0.5} = 4$ clock cycle

$$t_{cpu} = \frac{2}{\frac{1}{f}} = 2 \times f = \text{single operation time} \times \text{frequency}$$

$$t_{cpu} = 2 \times \frac{1}{T} = \frac{2}{T} = \frac{\text{single operation time}}{\text{Time Period}}$$



$\longrightarrow$ $2\ \mu sec$

⊛ t_{cpu} = No unit, only number

= cycles per instruction

= CPI

Therefore,

CPI for addition is 4.

⊛ t_{cpu} = single operation time × frequency

$$= \frac{\text{single operation time}}{\text{Time Period}}$$

⊛ Division operation required 12 clock cycle

or,

CPI for division is 12.

Given frequency, f = 2 MHz

⇒

$$\text{Time period, } T = \frac{1}{f} = \frac{1}{2\times10^{6}} = 0.5 \times 10^{-6} \text{ sec}$$

$$= 0.5\ \mu sec$$

∴ time for division $= (0.5 \times 12)\ \mu sec$

$= 6\ \mu sec$

Ans

⊛ Program in Assembly Language

		CPI
LOAD	R1, A →	4
ADD	R1, B →	3
ADD	R1, C →	3
MOV	R2, R1 →	2
OUT	25, R2 →	8
	Total CPI =	20

∴ Average CPI = $\frac{20}{5}$

$= 4$

⊛ we can calculate average CPI of a program not exact CPI on time. Because modern processor do multiple operation at once.

⊛

Instruction Type	Instruction Count	CPI
ALU	500	2
LOAD	200	4
STORE	200	4
BRANCH	100	6

Average CPI = $\frac{500 \times 2 + 200 \times 4 + 200 \times 4 + 100 \times 6}{1000}$

=

* From Lecture-2 → slide page 14

$$\text{Total Run time} = 1.55 \times 100000 \times T$$

$$= 1.55 \times 100000 \times \frac{1}{f}$$

$$= 1.55 \times 100000 \times \frac{1}{40 \times 10^6}$$

L-6/07.08.2023

* Program

Types	I_i Instruction Count		CPI_i
$i=1$ ALU	400	$= \frac{400}{900} = 0.44$	2
$i=2$ LOAD	300	$= \frac{300}{900} = 0.33$	6
$i=3$ BRANCH	200	$= \frac{200}{900} = 0.22$	4

Alternative Way

$$\text{Average CPI} = \frac{400 \times 2 + 300 \times 6 + 200 \times 4}{900}$$

$$= \frac{\sum_{i=1}^{n} (I_i \times CPI_i)}{\sum_{i=1}^{n} I_i}$$

$$= \frac{\sum (I_i \times CPI_i)}{I}$$

total instruction count of a program (without sub script)

⊛ Only fraction is given, then

$$\text{Average } CPI = \frac{I_1}{I} \times CPI_1 + \frac{I_2}{I} \times CPI_2 + \frac{I_3}{I} \times CPI_3$$

$$= \text{Fraction} \times CPI_1 + F_2 \times CPI_2 + F_3 \times CPI_3$$

$$= \sum_{i=1}^{n} (F_i \times CPI_i)$$

$$= 0.44 \times 2 + 0.33 \times 6 + 0.22 \times 4$$

$$= 3.74$$

⊛ Percentage given, then,

(i) Can divide by 100 every percentage to get the fraction.

(ii) Ignore the percentage sign and assume that $I = 100$.

⊛ First Distribute ~~and~~ of instruction given then,

$f = 2$ MHz

$T_{CPU} = ?$ = how much ~~the~~ time needed to run the program?

= Total number of Clock Cycles of all Instruction of the program × Time Period.

$$= (400 \times 2 + 300 \times 6 + 200 \times 4) \frac{1}{f} \rightarrow 0.5 \mu sec$$

$$\frac{1}{2 \times 10^6} = 0.5 \times 10^{-6}$$

$$= 0.5 \mu sec$$

$$= (3400 \times 0.5 \times 10^{-6}) sec$$

$$= 1700 \times 10^{-6} sec$$

$$= 1700 \mu sec$$

⊛ For calculate total run time of a program, we must know the I = total instruction count.

Total number of Clock Cycle = $I \times$ Average CPI

$T_{CPU} = I \times$ Average CPI $\times C \rightarrow$ time period

→ time for one cycle.

Average CPI → Convention

$$= I \times CPI \times C$$

$$= 900 \times 3.74 \times 0.5 \times 10^{-6}$$

$$= 1683 \mu sec$$

⊛ T_{CPU} for ALU $= T_{ALU} = 400 \times 2 \times C$

$$= I_{ALU} \times CPI_{ALU} \times C$$

$$T_{LOAD} = I_{LOAD} \times CPI_{LOAD} \times C$$

$$T_{BRANCH} = I_{BRANCH} \times CPI_{BRANC} \times C$$

$$\frac{T_{ALU}}{T_{CPU}} = \frac{I_{ALU} \times CPI_{ALU} \times C}{I \times \text{Average } CPI \times C}$$

$$= \frac{I_{ALU}}{I} \times \frac{CPI_{ALU}}{\text{Avg } CPI_{ALU}}$$

Fraction of Instruction

⊛ MIPS

M → millions

IPS → Instruction Per Second

$I = 900$

$T_{CPU} = 1700 \ \mu sec$

$$\therefore IPS = \frac{I}{T_{CPU}} = \frac{900}{1700 \ \mu sec}$$

$$MIPS = \frac{I}{T_{CPU} \times 10^6}$$

MIPS ⇒ Can process 15 millions instruction per second.

$$MIPS = \frac{\cancel{I}}{\cancel{I} \times \text{Avg } CPI \times C \times 10^6}$$

$$= \frac{f}{\text{Avg } CPI \times 10^6}$$

⊛ M FLOPS

M → millions

FLOPS → Floating point operation per second

$A = 1.56 \times 10^9$
$B = 1.35 \times 10^3$

± Hard, if it is not same exponent

× ÷ easy

Quiz-1 ⇒ Up to this.

L-7 / 07.08.2023 /

M FLOPS → Floating point operation per second

$A = 1.65 \times 10^5$
$B = 1.36 \times 10^9$

Floating point Unit (FPU)

Math Co-processor

$A = 16$
$B = 15$

Integer ALU

Amdahl's Law:

Run time

OLD System		New System
Input/Output ⇒ 20	⟶	20
Memory ⇒ 50	⟶	25
ALU ⇒ 30	⟶	30
100 sec		75 sec

$$\text{Speed Up, } S = \frac{\text{Old System time}}{\text{new system time}} = \frac{100}{75} = 1.33$$

memory ≅ 0, then run time of new system

$$S = \frac{100}{50} = 2$$

* 30
  $20 \longrightarrow 0$
  50
  100   80

$$S = \frac{100}{80} = 1.25$$

* 10
  $80 \longrightarrow 5$
  10
  100   25

$$S = \frac{100}{25} = 4$$

⊛ $20 = 0.2$ → 20

$50 = 0.5$ ⇒ 25 → $\frac{50}{25} = 2 = S$ for memory only

$30 = 0.3$ → 30

$100 = 1$

Overall speed up, $S = \dfrac{1}{(0.2+0.3) + \frac{0.5}{2}}$

$= \dfrac{1}{0.2+0.3+0.25}$ — time on old system in normalize form

$= \dfrac{1}{0.75} = 1.33$

$= \dfrac{1}{(1-F) + \frac{F}{S}}$

S → speed up of affected part's

F → affected fraction

$(1-F)$ → unaffected fraction

for memory

$S_{max} = ?$

$= \dfrac{50}{0} = \infty$

$$\therefore \text{Speed up maximum} = \frac{1}{(1-0.5) + \frac{0.5}{\infty}} \quad \frac{0.5}{\infty} \rightarrow \cong 0$$

$$= \frac{1}{0.5} = 2$$

⊛ Amdahl's Law Proved

$$\boxed{\text{Speed up} = \frac{1}{(1-F) + \frac{F}{S}}}$$

L-8/14.08.2023/

$$\text{Avg CPI} = 0.6 + 1.8 + 1.0 + 0.8 = 4.2$$

$$\text{MIPS} = \frac{1}{T_{cpu} \times 10^6} = \frac{I}{I \times \text{Avg CPI} \times c \times 10^6} = \frac{f}{\text{Avg CPI} \times 10^6}$$

$$= \frac{2.3 \times 10^9}{4.2 \times 10^6}$$

$$= \frac{2.3}{4.2} \times 10^3$$

$$= 0.55 \times 10^3$$

$$= 550$$

fraction

(*) $$T_{STRING} = \frac{I_{STRING} \times CPI_{STRING} \times \cancel{c}}{I \times \text{Avg CPI} \times \cancel{c}}$$

$$= 0.2 \frac{5}{4.2} = 0.24$$ processing time for a particular instruction in normalized form.

$$\text{Speed Up} = \frac{\text{Old System Run time}}{\text{New System Run time}} = \frac{\cancel{I} \times \text{Avg CPI} \times \cancel{c}}{\cancel{I} \times \text{Avg CPI} \times \cancel{c}}$$

$$= \frac{4.2}{3.6} = 1.17$$

$$= \frac{1}{(1-F) + \frac{F}{S}}$$ Accelaration fraction

$$S = \frac{5}{2} = 2.5$$

$$= \frac{1}{(1-0.24) + \frac{0.24}{2.5}} = \dots$$

$F\uparrow$, S = Fixed
Then, Speed Up $\uparrow$

Max Speed Up = $\frac{1}{(1-1) + \frac{1}{2.5}} = 2.5 = S$

F = 1

⊛ Amdahl's Law:

$$\frac{1}{(1-\Sigma F_i) + \Sigma \frac{F_i}{S_i}}$$

$$= \frac{1}{(1-F_1-F_2-F_n) + \frac{F_1}{S_1} + \frac{F_2}{S_2} + \frac{F_n}{S_n}}$$

$T_{LOAD} = \frac{0.3 \times 6}{4.2} = \frac{1.8}{4.2} = \cdots$

⊛ If CPI for each instruction is same, then the fraction of instruction and ~~ti~~ time will be the same

⊛ If CPI for each instruction is not given, then fraction of instruction will decide the fraction of time of that particular instruction.

Parallel System

Task

15 problems ⇒ Each problems takes 2 sec

5 problems sequential

10 problems parallel

$$p = \frac{10}{15} = \frac{2}{3}$$

→ fraction of work can be done in parallel.

$$\text{Speed Up} = \frac{1}{(1-p) + \frac{p}{n}}$$

n → number of processor

L-9/16.08.2023/

$$MIPS = \frac{I}{T_{cpu} \times 10^6} = \frac{\cancel{I}}{\cancel{I} \times Avg\ CPI \times C \times 10^6}$$

$$= \frac{f}{Avg\ CPI \times 10^6}$$

$$= \frac{250 \times 10^6}{2.7 \times 10^6} = \ldots$$

⊛ ~~Ex~~ Execution time is more important than MIPS.

c)

$f = 200\ MHz$

$$T = \frac{1}{200} \times 10^{-6} = 0.005\ \mu sec$$

$$= 0.005 + \frac{0.005 \times 20}{100}$$

$$= (0.005 + 0.001)\ \mu sec$$

$$= 0.006 \times 10^{-6}\ sec$$

$$f = \frac{1}{T} = \frac{1}{0.006 \times 10^{-6}}\ Hz = 166.66\ MHz$$

$$\frac{T_{LOAD}}{T_{cpu}} = \frac{I_{LOAD} \times CPI_{LOAD} \times C}{I \times Avg\ CPI \times C} = \frac{0.3 \times 5}{4.75} = 0.32 = F_1$$

$$\frac{T_{BRANCH}}{T_{CPU}} = \frac{0.15 \times 6}{4.75} = 0.189 = F_2$$

$$\text{Speed Up} = \frac{1}{(1 - 0.32 - 0.189) + \frac{0.32 \times 3}{5} + \frac{0.189}{2}}$$

$$= \ldots$$

80% faster,

Assume thats, Run time on New = 100 sec

" " " Old = (100 + 80) sec

= 180 sec

$$\therefore S = \frac{180}{100} = 1.8$$

Next Wednesday
Mid-1

L-10/21.08.2023/

Some Class Practice

L-11/23.08.2023/

Program

.	80% memory	20% CPU

S

60% of memory operation is accelarated by 2 times

$$F = 0.8 \times 0.6 = 0.48$$

$$\therefore \text{Speed up} = \frac{1}{(1-0.48) + \frac{0.48}{2}}$$

⊛ Parallel System Problem.

for max

$$\text{Speed up} = \frac{1}{(1-P) + \frac{P}{n}} = \frac{1}{(1-0.6) + \frac{0.6}{\infty}}$$

$$= \frac{1}{0.4} = 2.5$$

⊛

old time

$$2.5 - \left(100 \times 10^6 \times \frac{40}{100} \times 5 \times \frac{1}{800 \times 10^6}\right) \times \frac{20}{100}$$

$$= 2.45$$

new time

$$\therefore \text{Speed up} = \frac{2.5}{2.45} = 1.020$$

⊛ Stack ⇒ a ~~patta~~ part of a memory

AC

⊛ Accumulator based Processor:

LOAD M1 [Destination is implicit as its predefined to load in AC]

STOR M2 [Source is implicit]

⊛

LOAD M1; $AC \leftarrow [M1]$

ADD M2; $AC \leftarrow [M2] + AC$

STOR M3; $[M3] \leftarrow AC$

Next class Mid