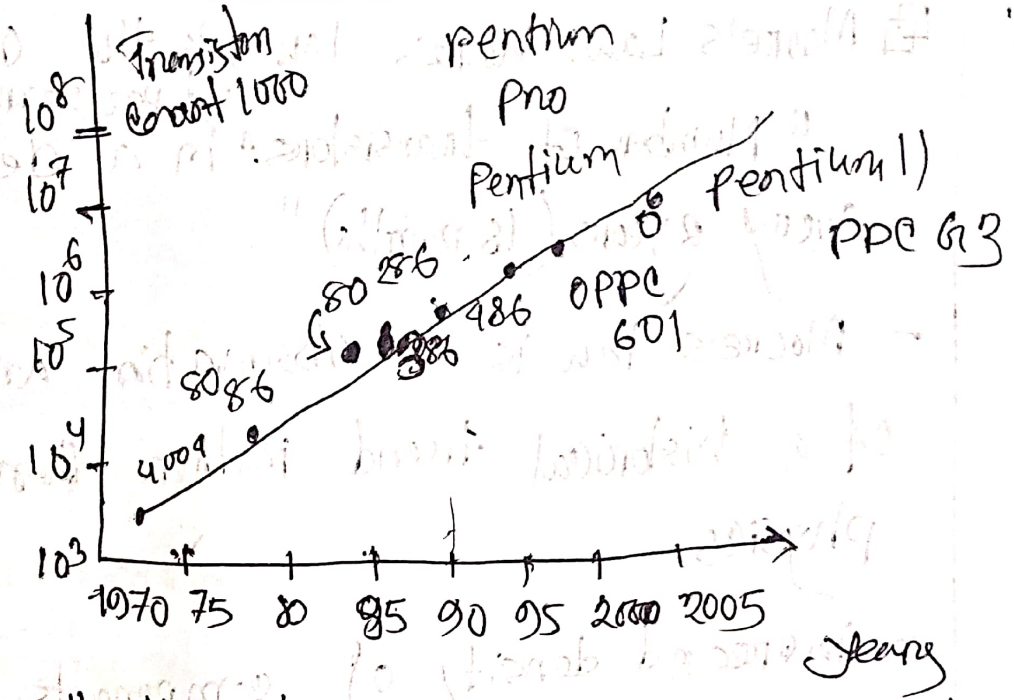
Computer Architecture and Organization

- Moore's Law: Moore's law is the observation that "Number of transistors used per square inch in a dense IC's double every 2 years (18 months)"
  - Moore's law is an observation and projection of a historical trend rather than a law of physics.
  - Increased density of components on chip
  - Gordon Moore - Cofounder of Intel
  - Num of transistors on a chip will double every year.
  - Cost of a chip has remained almost unchanged
  - Higher packing density means shorter electrical paths, giving higher performance
  - Smaller size gives increased flexibility
  - Reduced power and cooling requirements
  - Fewer interconnections increases reliability

The number of trans on IC chips



- As a result the scale gets smaller & smaller & the performance of microprocessor has enjoyed an exponential growth.
- In 1970 there were around 10^3 transistors placed in an integrated chip.
- Whereas in 2010 there were nearly 1 billion transistors were placed in a chip & still the researches are going on to increase the number
- The moore's law plays a major role in the VLSI Circuits. This tiny

electronic circuit consists of num IC's and they in turn possess many transistors which supports the circuitry.

* facts:

- Misconception: Reduction of transistor gate length less than 22 nm is not possible.
- New Inventions: Recently INTEL has reduced transistor gate length less than 22nm.
- Result: The invention indicates that Moore's
  - law road doesn't end
  - As long as the thirst for innovation is present Moores law trend will continue.

Advancements: The contribution of Moore's law has given rise to the reduction in the robust structure of the devices to handy gadgets over these decades.

2.9 The ENIAC was a decimal machine, where a register was represented by a ring of 10 vacuum tube. At any time, only one vacuum tube was in the ON state; representing one of the 10 digits. Assuming that ENIAC had the capability to have multiple vacuum tubes in the ON & OFF state simultaneously, is this representation useful? and what range of int values could we represent using the 10 vacuum tubes?

Ans:

Step 1 - The electronic numerical integrator and computer, or ENIAC, is regarded as the country's first operating electronic digital computer. The ENIAC, which was completed in 1945, was the first programmable, electronic, general-purpose digital computer. These features were available on

other computers, but the ENIAC offered them all in one convenient package. It could solve a vast class of numerical problems through reprogramming and was Turing-complete. It's interesting to note that the ENIAC was decimal, not binary! The digits 0 through 9 were represented by 10 vacuum tubes.

steps 2:

Each day or two, ENIAC lost one vacuum tube. It was difficult to find and replace the broken tube because there were over 18,000 of them. However, the maintenance crew eventually honed the ability to solve a problem in just 15 minutes. this format is inefficient because it takes ten tubes to express just one decimal digit from 0 to 9. these same tubes could be

used as binary bits if any number of them could be turned on simultaneously. These patterns can be used to represent integers ranging from 0 to 1023.

* Using binary representation, here's how the 10 decimal values (0 to 9) could be represented with 4 vacuum tubes (bits):

Decimal: 0 1 2 3 4 5 6 7 8 9

Binary : 0000 0001 0010 1001

2.10

Consider a machine with 40 MHz processor which has run a benchmark program. The executed program consists of 100,000 instruction executions, with the following instruction mix and clock cycle count. What will be the effective CPI, MIPS rate,

and execution time.

Instruction type	Instruction Count	Cycles/Instructions
Integer arithmetic	95000	1
Data Transfer	32000	2
Floating point	15000	2
Control transfer	8000	2

Ans:

Given, frequency = 40 MHz

Clock time = $\frac{1}{40 \times 10^6}$ sec

Ins type	Ins Count	Cycle/Ins	% of ins type
Integer arithmetic	95000	1	95%
Data transfer	32000	2	32%
Floating point	15000	2	15%
Control Transfer	8000	2	8%
Total	1060000		100%

$$CPI = \sum_i (\text{fraction of instruction type } i \times \text{cycles needed by instruction type } i)$$

$$\Rightarrow CPI = 0.45 * 1 + (0.32 + 0.08 + 0.15) * 2$$

$$= 1.55.$$

Execution time = CPI * total num of ins * clock time.

$$= 1.55 \times 10^5 \times \frac{1}{40 \times 10^6} \text{ sec}$$

$$= 3.875 \text{ ms}$$

$$MIPS = \frac{\text{clock frequency}}{CPI * 10^6}$$

$$= \frac{40 \times 10^6}{1.55 \times 10^6} = 25.8$$

2.13 Four benchmark programs are executed on three computers with the following results;

	Computer A	Computer B	Computer C
Program 1	1	10	20
Program 2	1000	100	20
Program 3	500	1000	50
Program 4	100	800	100

the table shows the execution time in seconds with 100,000,000 instructions executed in each of the four programs. Calculate the MIPS values for each computer for each program.

Ans: MIPS rate given as $MIPS = \frac{I_c}{T \times 10^6}$

Arithmetic mean is $R_A = \frac{1}{m} \sum_{i=1}^{m} R_i$

Harmonic mean is $R_H = \frac{m}{\sum_{i=1}^{m} \frac{1}{R_i}}$

By applying MIPS $= I_c / (T \times 10^6)$

$= 100,000,000/(1 \times 10^6) = 100/T$. Therefore, the MIPS values are:

	Computer A	Computer B	Computer C
Program 1	100	10	5
Program 2	0.1	1	5
Program 3	0.2	0.1	2
Program 4	2	0.125	1

	Arithmetic mean	Rank
Computer A	25.575	1
Computer B	2.80	3
Computer C	3.25	2

	Harmonic mean	Rank
Computer A	0.25	2
Computer B	0.21	3
Computer C	0.1	1

... memory we / we

Mid-1

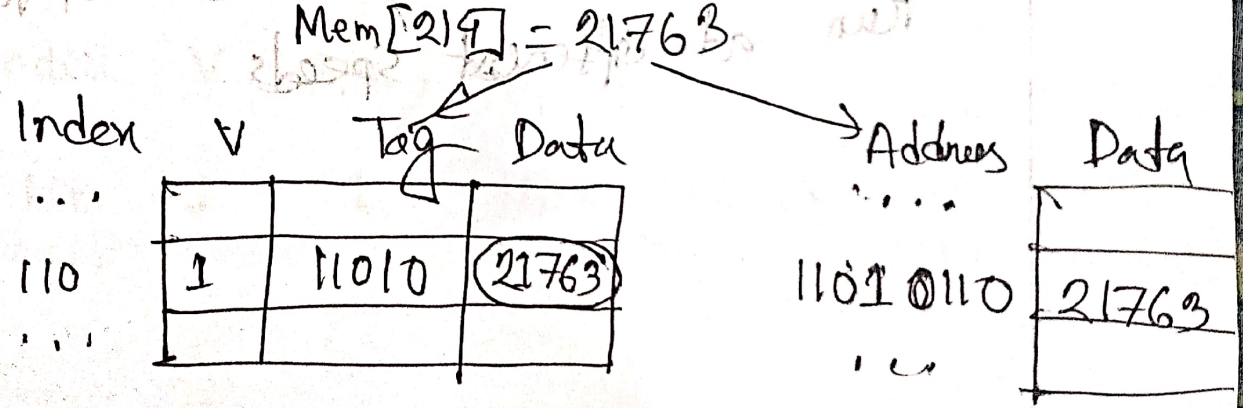
1. ENIAC

2. When a block that is resident in the cache is to be replaced, there are two cases to consider; describe the write through and write back.

Ans:

Write through caches:

A write through cache solves the inconsistency problem by forcing all writes to update both the cache and the main memory.

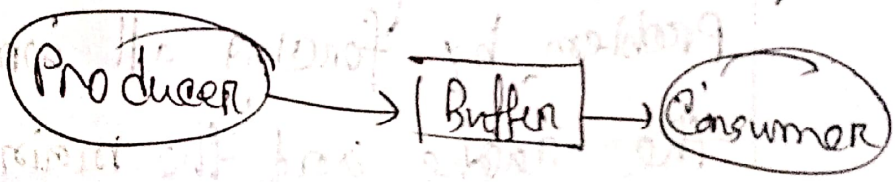


This is simple to implement and keeps the cache and memory consistent.

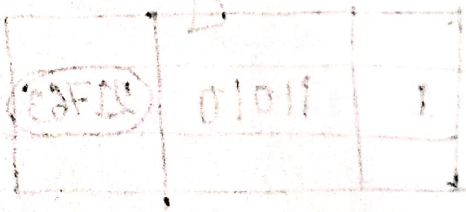
The bad thing is that forcing every write to go to main memory, we use

up bandwidth between the cache and the memory.

Write through caches can result in slow writes, so, Processors typically include a write buffer, which queues pending writes to main memory and permits the CPU to continue.



Buffers are commonly used when two devices run at different speeds.



Write back: In a write back cache, the memory is not updated until the cache block needs to be replaced (when loading data into a full cache set).

For example, we might write some data to the cache at first, leaving it inconsistent with main memory as shown before.

- The cache block is marked "Dirty" to indicate this inconsistency.

Mem[21763] = 21763

Index	V	Dirty	Tag	Data
...				
110	1	1	11010	21763
...				

Addr.	Data
1000 1110	1225
1101 0110	42803

3. Consider a two level cache with access time 5 ns and 80 ns respectively. If the hit ratio is 95% and 75% respectively in the two caches & main memory access time is 250 ns. What is the effective access time?

We know,

Effective access time = (hit ratio) * (memory access time + search time in associative registers) + (fail ratio) * (search time in associative registers + 2 * memory access time)

⇒ $T_1 = 5ns$, $T_2 = 80ns$, $T_3 = 250 ns$

$H_1 = 0.95$, $H_2 = .75$.

$T_{Avg} = H_1 T_1 + (1-H_1) H_2 (T_1+T_2) + (1-H_1)(1-H_2)(T_1+T_2+T_3)$

Since search tion within a cache is ignored

$T_{Avg} = H_1 T_1 + (1-H_1) H_2 (T_2) + (1-H_1)(1-H_2)(T_3)$

$= 0.95 \times 5 + (1-0.95) \times 0.75 \times 80 + (1-0.95)(1-0.75) \times 500$

$= 4.75 + 3 + 3.125$

$= 11.375$ ns $= 10.875$

$T_{EAT} = 4.75 + (1-0.95) \times 0.75 \times (5+80) + (1-0.95)(1-0.75)(5+80+500)$

$= 4.75 + 0.5625 + 6.4375$

$= 11.75$

Q. Suppose three interrupt handlers A, B and C (having priority level A > B > C) with Interrupt Service Routine (ISR) 10, 30 & 20 respectively. Graphically show the transfer of control for interrupt sequence of C, A & B at time t = 10, 25 & 95 respectively.