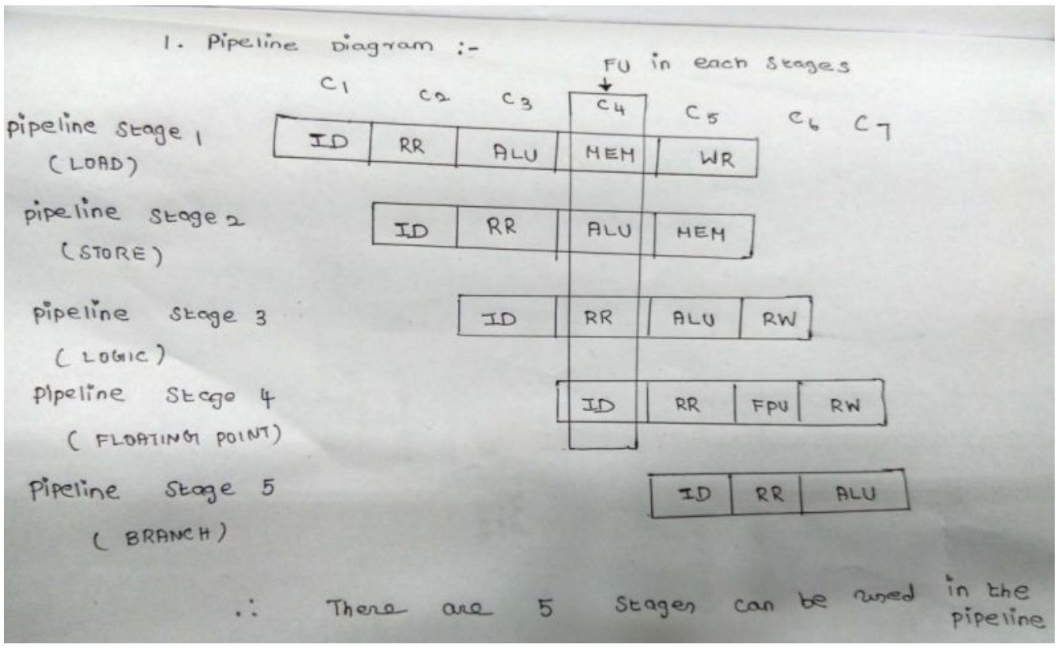


The CPU should be pipelined as below to support the all the six instruction types.

ID	WR	RR	ALU	FPU	MEM	WR

Number of stages:
There 7 stages are available in the processor.

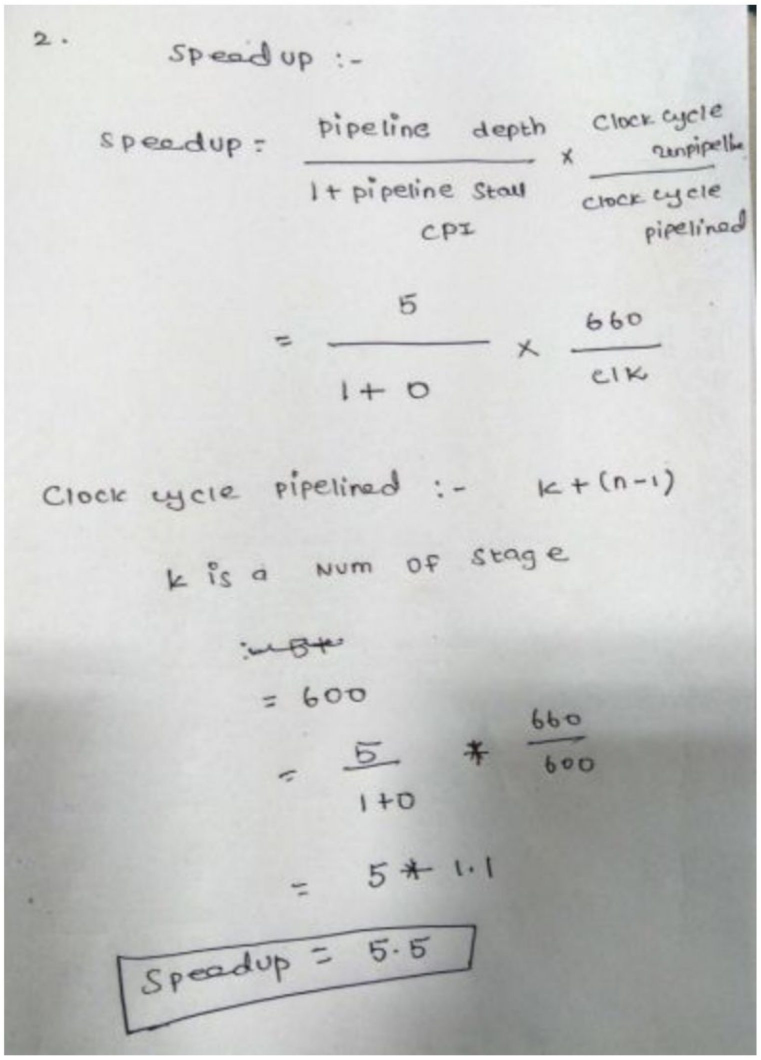
Operations in each stage:
ID – fetches the instruction & decodes it
WR – write the registers
RR – Read the registers
ALU – perform the arithmetic/logic operations
FPU – perform floating point operations
MEM – performs the memory access
WR – write the registers

New Clock rate:
Clock rate = 1/ 1 cycle time
= 1/280 ps
= 3.5 GHz

Comparing speed up with real 660 ps instruction:
Time needed to execute the single instruction = 280 ps (longest stage)

Speed up = 660 / 280
= 2.35

If the number of registers is doubled:
There is no need to make changes in the pipelined machine.
When the registers count is doubled and adding extra time to the registers, never affect the clock or stages of the pipelined-processor.



2. Speedup :-

$$\text{speedup} = \frac{\text{pipeline depth}}{1 + \text{pipeline stall CPI}} \times \frac{\text{Clock cycle unpipelined}}{\text{clock cycle pipelined}}$$

$$= \frac{5}{1+0} \times \frac{660}{\text{clk}}$$

Clock cycle pipelined :- $k+(n-1)$

k is a Num of stage

[illegible]

$= 600$

$$= \frac{5}{1+0} * \frac{660}{600}$$

$= 5 * 1.1$

$\boxed{\text{Speedup} = 5.5}$

Given:
L1 cache
256K
Read hit rate = 90 %
Miss rate = 10%
miss penalty = 4 clock cycles

L2 cache
=2MB
hit rate = 99.5%
miss rate = 0.5 %
miss penalty = 60 clock cycles

Memory access time =L1+L2 miss penality= 4+ 60 = 64 cycles

1.
CPI = 1 + percentage miss in L1 * (L2access-time+percentage miss in L1 * Memory access-time)
= 1+ 10% x (60 + 0.5 %x 64)
= 1+ 0.1 x (60 +0.32)
= 1+ 6.032
CPI = 7.032 cycles
Since write and read instructions 40 % frequency , the remaining 60 % 1 clock cycle
Thus CPI = 7.032 + 1 =8.032

2)
Now L1 cache hit rate is changed to 95 %
clock rate = 2.2 Ghz
CPI = 1 + percentage miss in L1 * (L2access-time+percentage miss in L1 * Memory access-time)
= 1+ 5% x (60 + 0.5 %x 64)
= 1+ 3.016
CPI = 4.016 cycles
Since write and read instructions 40 % frequency , the remaining 60 % 1 clock cycle
Thus CPI = 4.016 + 1 = 5.016

performance
= 1st CPI/ 2nd CPI * (2.2/2.4)
=1.601 x 0.916
= 1.467
= 1+ 0.467
This means that there is 46.7% improvement that is worthwhile

Given

L1 cache hit rate = 90 %

miss rate in L1 = 10 %

L2 cache hit rate = 98 %

miss rate in L2 = 2 %

Cache L1 miss penalty = 4 clock cycles

Cache L2 miss penalty = 30 clock cycles

Memory access time = 4+ 30 = 34 cycles

Solution

1)

formula

CPI = 1 + L1 miss x (L2access-time

+ L2 miss x Memory access-time)

CPI = 1+ 10% x (30 + 2 % x 34)

= 1+ 0.1 x (30 +0.68)

= 1+ 0.1 x (30.68)

= 1+ 3.068

CPI = 4.068 cycles

Remaining 30 % percent instructions takes 1 clock cycle

Given

L1 cache hit rate = 90 %

miss rate in L1 = 10 %

L2 cache hit rate = 98 %

miss rate in L2 = 2 %

Cache L1 miss penalty = 4 clock cycles

Cache L2 miss penalty = 30 clock cycles

Memory access time = 4+ 30 = 34 cycles

Solution

1)

formula

CPI = 1 + L1 miss x (L2access-time

+ L2 miss x Memory access-time)

CPI = 1+ 10% x (30 + 2 % x 34)

= 1+ 0.1 x (30 +0.68)

= 1+ 0.1 x (30.68)

= 1+ 3.068

CPI = 4.068 cycles

Remaining 30 % percent instructions takes 1 clock cycle

There fore CPI

= 4.068 + 1

=5.068

2)

hit rate = 94 %

Initial clock rate = 2.4 Ghz

now clock rate = 6 cycles

CPI = 1 + L1 miss x (L2access-time

+ L2 miss x Memory access-time)

CPI = 1+ 2% x (30 + 2 %x 34)

= 1+ 0.02 x (30 +0.68)

= 1+ 0.02 x (30.68)

= 1+ 0. 6136

CPI = 1. 6136 cycles per instruction for read / write operations

Remaining 30 % percent instructions takes 1 clock cycle

There fore CPI

= 1. 6136 + 1

= 2. 6136

performance ratio

= (5.068/ 2. 6136) x (6/2.4)

=1.93 x 2.5

= 4. 825

Answer:

We have given , Virtual address = 48 bits

Real memory = 32GB

16KB pages

real memory = 32GB = 2^5 x 2^30 B = 2^35 B

Therefore, bits in a real address = 35

Bits in virtual address = 48 bits

Now pages = 16 KB = 2^4 x 2^10 B = 2^14 B

Now , pages = real address - page offset

= 35 - 14

= 21

Number of pages = 2^21

Similarly frames = 2^48

Therefore page table entry = 48

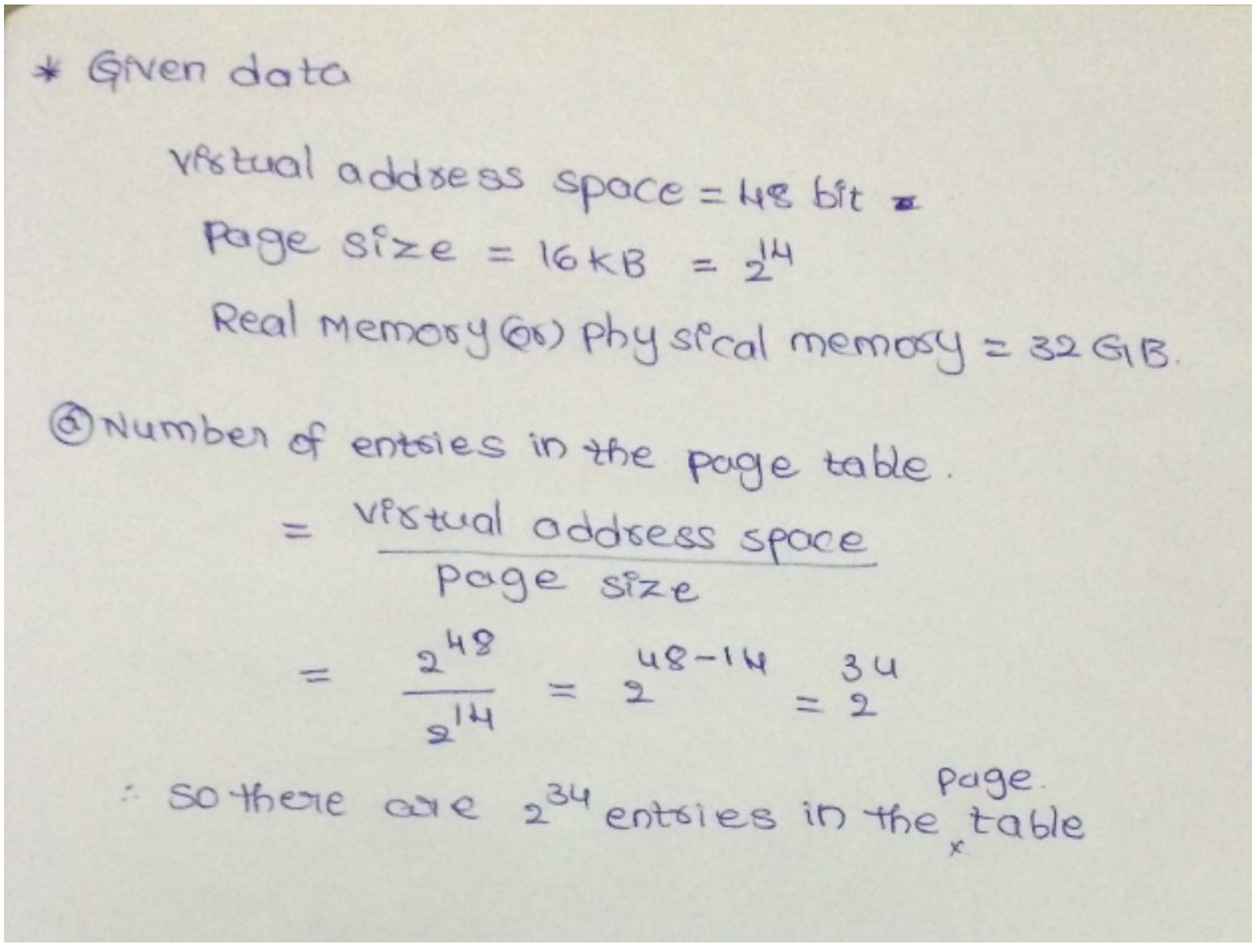
Since we need not to select a page for replacement , so not required

Page Size = 16KB = 2^4 x 2^10 B = 2^14 B

Therefore Page offset = 14 bits

For 8 byte page table entry, we need 2^48 x 8 B

We have the concept of virtual memory , which increases the degree of multi programming to avoid such complexities.



* Given data

virtual address space = 48 bit

Page size = 16 KB = 2^{14}

Real memory (or) physical memory = 32 GB.

ⓐ Number of entries in the page table.

$$= \frac{\text{virtual address space}}{\text{page size}}$$

$$= \frac{2^{48}}{2^{14}} = 2^{48-14} = 2^{34}$$

∴ So there are 2^{34} entries in the page table

Each block is going to map to subset of cache

2 way associative = 2 blocks in set i.e 4 block/2 = 2 sets. Need to provide index equal to 1 bit.

Here 256 entries are provided so there will be 256/2 = 128 sets.

tag	Set number	Byte offset
20	8	4

31 0

Main Memory address for 32 bit.

- More than one memory block is mapped onto the same

position in the cache

- Memory address is divided into three fields:

  - Low order 4 bits determine one of the 16

    words in a block.

When a new block is brought into the cache,

the the next 8 bits determine which cache

block this new block is placed in.

High order 20 bits determine which of the possible

128 blocks is currently present in the cache. These

are tag bits.