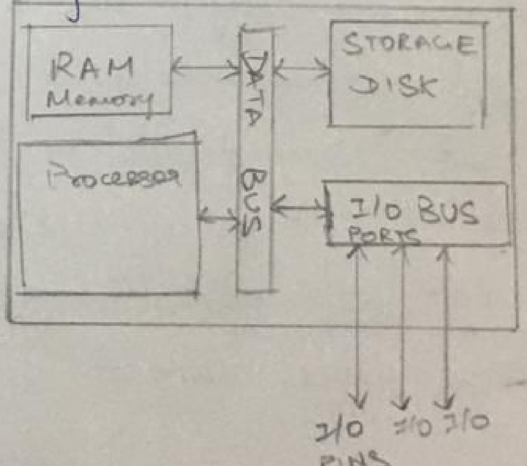
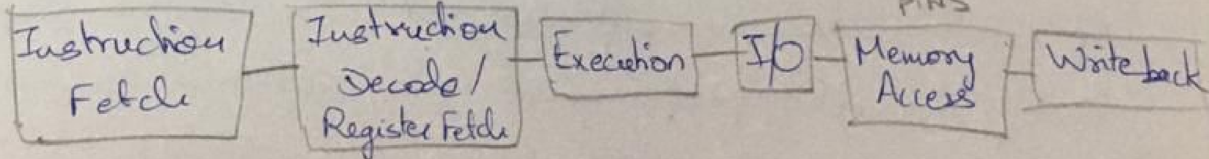
N. KAUSIK
COE17B010
N. Kus
A. Naray

Computer Architecture EndSem

1. a) Given that processor needs to do integer, floating point shift and logical operations.
I/O processing should be ONCHIP and I/O pins are placed externally.
So, designed as,
3 components (Processor, Memory, Storage) connected via Bus.
External I/O ports connected to I/O Bus which is connected to Data Bus.
I/O ports are connected parallelly to Data Bus via I/O Bus providing multiple ports to communicate to chip simultaneously. I/O Bus and Data Bus operate on different clock cycle times due to I/O being much slower.



- Processor is designed for pipelined execution with 6 stages.



To further improve, Inorder Issue, Out of Order Execution, Out of order Commit is used along with ~~Opera~~ Operator forwarding from output of Exe, I/O and Mem to input of

[illegible]
[illegible]

Exe.

- Memory on same chip.
  Standard DDR4 (Double Data Rate) DRAM used.
  2.66 GHz, 16 GB, 100 ns
- Storage on same chip.
  SSD (NVMe pCIe) can be used for fast access and smaller size (physical size). 256 GB space.

b) As processor is pipelined and out of order execution is used, this design will give higher throughput.

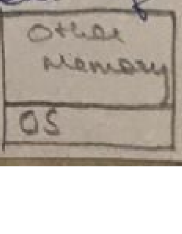
c) Superscalar processor is 5-stage pipelined and hence it waits for I/O.
My design has I/O included as a stage in pipeline itself.
So, my design has a higher cost as more hardware, but execution speed is higher than superscalar.
But throughput is lesser than superscalar due to extra stage in pipeline.

d) For 100 instructions,

$$\text{Performance Ratio} = \frac{100+6-1 \text{ (my Design)}}{100+5-1 \text{ (Superscalar)}} = \frac{105}{104} = 1.009$$

As I/O is also in pipeline, it is an extra stage, but overall performance gets improved as no more extra waiting for I/O like Superscalar.

e) It can be done by allocating a specific segment of memory only for OS and applications.



[illegible]
[illegible]

f) DDR4 - 2.66 GHz is used to store OS and applicati programs as it has fast access time, less latency and DDR (Double Data Rate).

g) Direct - Mapped Cache Memory is used.
Improvements are used like Critical Word First, Read miss priority and merging write buffers to reduce its miss penalty.
As direct-mapped, hit time is low.
Compiler optimization is used to further reduce miss rate.

[illegible]
[illegible]

2. Analysis - Assuming 50% Miss Rate and Intel Core i7 Cache Hit times

a) As only flash memory,

perbit Cost ⇒ \$ 0.0031 / MB = \$ 3.695×10^{-10} per bit

As processor exec time is negligible,

Exe time = 10 ms

b) On chip cache and flash memory

perbit Cost ⇒ $\underset{(cache)}{\$10/MB}$ + $\underset{(flash)}{\$0.0031/MB}$

= \$ 1.192×10^{-6} per bit

Exe time = $\overset{(onchip)}{40\text{ ns}} + 0.5 \times \overset{(flash)}{10^{7}\text{ ns}}$

= 5000040 ns = 5.00004 ms

c) 1 on chip cache, 1 off chip cache, flash memory

perbit Cost ⇒ \$ 10/MB × 2 + \$0.0031/MB

= \$ 2.384×10^{-6} per bit

Exe time = $\overset{(onchip)}{40\text{ ns}} + 0.5 \times (\overset{(offchip)}{160\text{ ns}} + 0.5 \times \overset{(flash)}{10^{7}\text{ ns}})$

= 40 + 80 + 2500000 = 2.50012 ms

d) L1 on chip, L2 offchip, flash memory

per bit Cost = \$ 2.384×10^{-6} per bit

Exe time = $\overset{(L1\ onchip)}{2\text{ ns}} + 0.5 \times (\overset{(L2\ offchip)}{5\text{ ns}} + 0.5 \times 10^{7}\text{ ns})$

= 2 + 2.5 + 2500000 = 2.5000045 ms

e) L1 onchip, L2 onchip, flash memory

per bit Cost = \$ 2.384×10^{-6} per bit

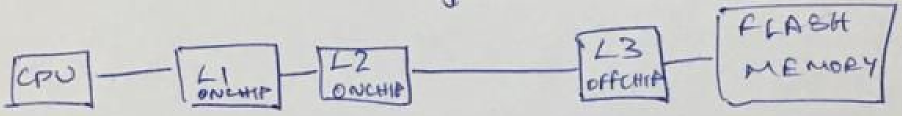
Exe time = $2\text{ ns} + 0.5 \times (3\text{ ns} + 0.5 \times 10^{7}\text{ ns})$

[illegible]
[illegible]

$= 2 + 1.5 + 2500000 = 2.5000035$ ms

So, from the analysis we can see that (e) performs best as having high speed L1 and L2 caches ON CHIP provides fastest access times but at more cost.

Better heirarchy,
we can add any extra L3 off chip cache to (e)



So, at extra cost,

Exe time $= 2 + 0.5(3 + 0.5(30 + 0.5 \times 10^7))$

$= 2 + 1.5 + 7.5 + 1250000$ ns

$= 1.2500011$ ms

So, in real scenarios it will take even less time as miss rates as generally very low and < 0.5.

N. Kw
A. Nag

3. a) As same number of pipeline stages, Type B VLIW has higher speed than Type A as B has multiple issue and A is only in-order issue.

b)

- Type A Limitations,
  i) Window size and issue count is Limited (upper bounded)
  ii) All constraints should be removed
  iii) Unit scheduling has more latency
  iv) More Hardware Structural complexity
  v) Realistic branch and jump prediction.
- Type B limitations,
  i) VLIW Decode Issue
  ii) Code size is more due to wasted fields and loop unrolling.
  iii) Hazards and Stalling of instructions
  iv) Due to varying no. of FU and latencies,

c) Both Type A and B support Multithreading.
Type A - Simultaneous Multithreading
Type B - Block interleaving
But it can cause problems like Spin Locks and thread Switching Overhead.
Also Type B Multithreading implementation is more complex and costly.

d) Type A -

Type B - used in System on Chips as it helps in pushing for less complex hardware and more complex Software.

N. Kur
A. Narog

e) - Type A Multi Core

i) Provides direct gain in perfomonce ($\propto$ no. of cores)

ii) More cores means more power consumption
  - Can be reduced using power gating

iii) Data Coherency Issues
  - Snooping / Directory protocol can be used to overcome

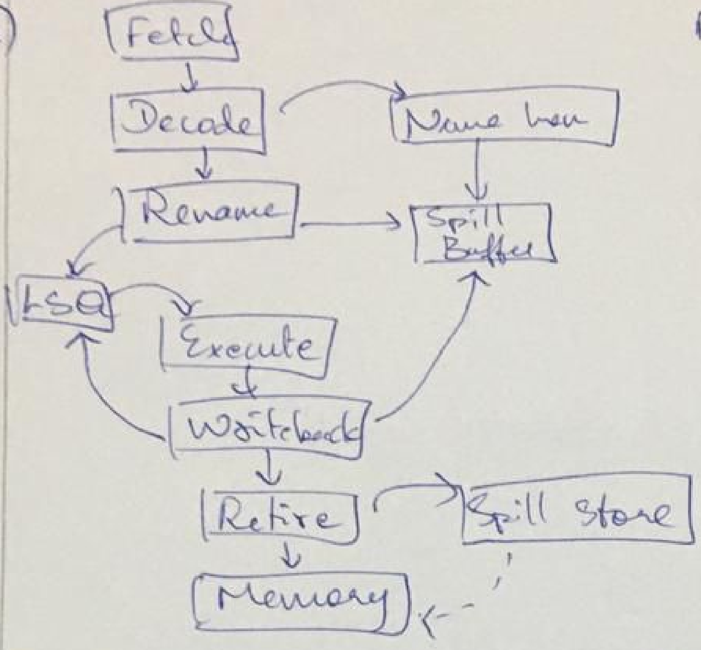
- Type B Multi Core

i) Not practical to use multiple cores as already made Very Long Instructions by grouping of ins.

ii) So, complexity of hardware becomes very high.

[illegible]

4.a)



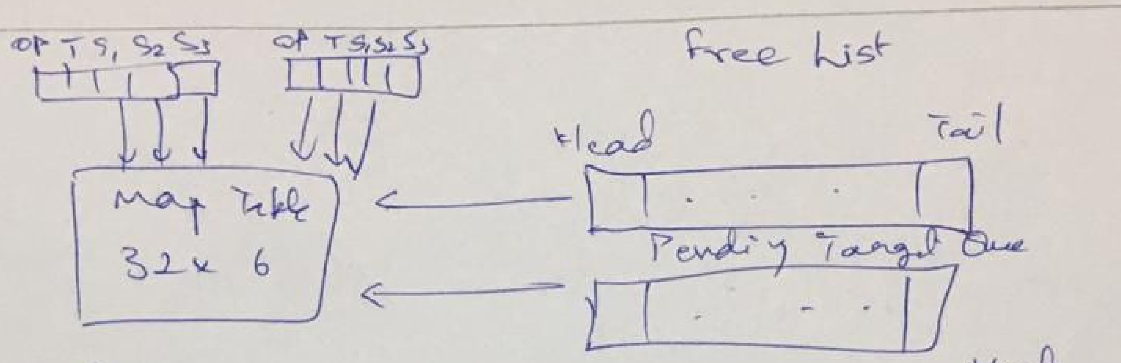
Architecture Pipeline

Store / Spill

Performance of Processor falls due to register spilling due to lack of available registers in it.
Great performance improvement and low power core. if we avoid register spilling by deciding value for no. of registers.

b) No. of registers for renaming
i) Size of ROB
ii) More ROB entries, more 'ins to be executed'
Ideal case, unlimited hardware resource for register renaming.
Solution,
i) Two separate register files
RRF ae 2 ARF
ii) Single large register pool
Pooled / merged / shared Reg file

N. Kew
A. Narq



Approaches,

i) Merged Reg File
ii) Architected Reg file + Rename Reg File.

[illegible]

4. d) Integer Multiplier Units cant be manufactured, but Adders, SDRAM Cells and Load/Store can be manufactured. So, the company can save money by using the same manufacturer and,
instead of using Integer Multiplier Units to perform multiplication, it can be done as repeated addition.

i.e. $\underbrace{12 \times 3}_{\text{cant be done}} = \underbrace{12 + 12 + 12}_{\text{can be done using adders}}$

So, the instruction,

MUL R1, R2, R3 can be rewritten as

```
Loop:
ADD R1, R1, R2
SUB R3, R3, #1
BNE Loop
```

and we can achieve same results.

So, whenever MUL instruction needs to be done, the (new) architecture will replace that by these 3 instructions. So, we get required result without incurring huge loss.

N. Kw
A. Nang

5. a) To support multi threading changes needed are,

i) Add thread load and store queues

ii) Increase L2 and L3 cache sizes

iii) Add separate instruction fetch and buffering for each thread

iv) Increase Virtual Register Count

v) L1 Cache Associativity increased for instructions and instruction address ~~transla~~ translation buffer

vi) Increase several issue queue size.

Eg. Physical size of POWER 5 (Given) is 24% more than POWER 4 due to Simultaneous Multithreading.

b) For out of order exec processors, with SMT, instruction from multiple threads are issued in exact same clock cycle. This hence uses register renaming and dynamic scheduling of multi issue architecture in CPU.
∴ This requires more hardware like extra register files, program counters in each thread and temp result registers used before commit are performed. More hardware is required to figure out mapping of threads and instructions.

Thus it increases processor execution speed ~~and~~ as utilisation of FUs is maximum.

c) Architectural trends have shifted from improving singly threaded application to ILP to improve multithreading application performance by supporting TLP.

N. Kw
A. Nag

Multicore processors having many cores in single die becomes ubiquitous. Multithreaded parallel programs has overhead due to communication b/w threads. Multithreading features are,
i) Resource Sharing
ii) Proximity of Application
iii) Data Sharing b/w Caches among processor's cores.