# Computer Architecture Endsem

Amar Kumar
CED17I029

5 (a) The architectural changes to support simultaneous multithreading efficiently for out of order execution

Simultaneous multithreading or SMT is one of the main implementation of multithreading which is better than any other type of multithreading such as fine or coarse grained multithreading.

In SMT, more than one thread can be executed in any given pipeline stage at a time

The main ability we need is that we could be able to fetch instructions from multiple threads in a cycle.

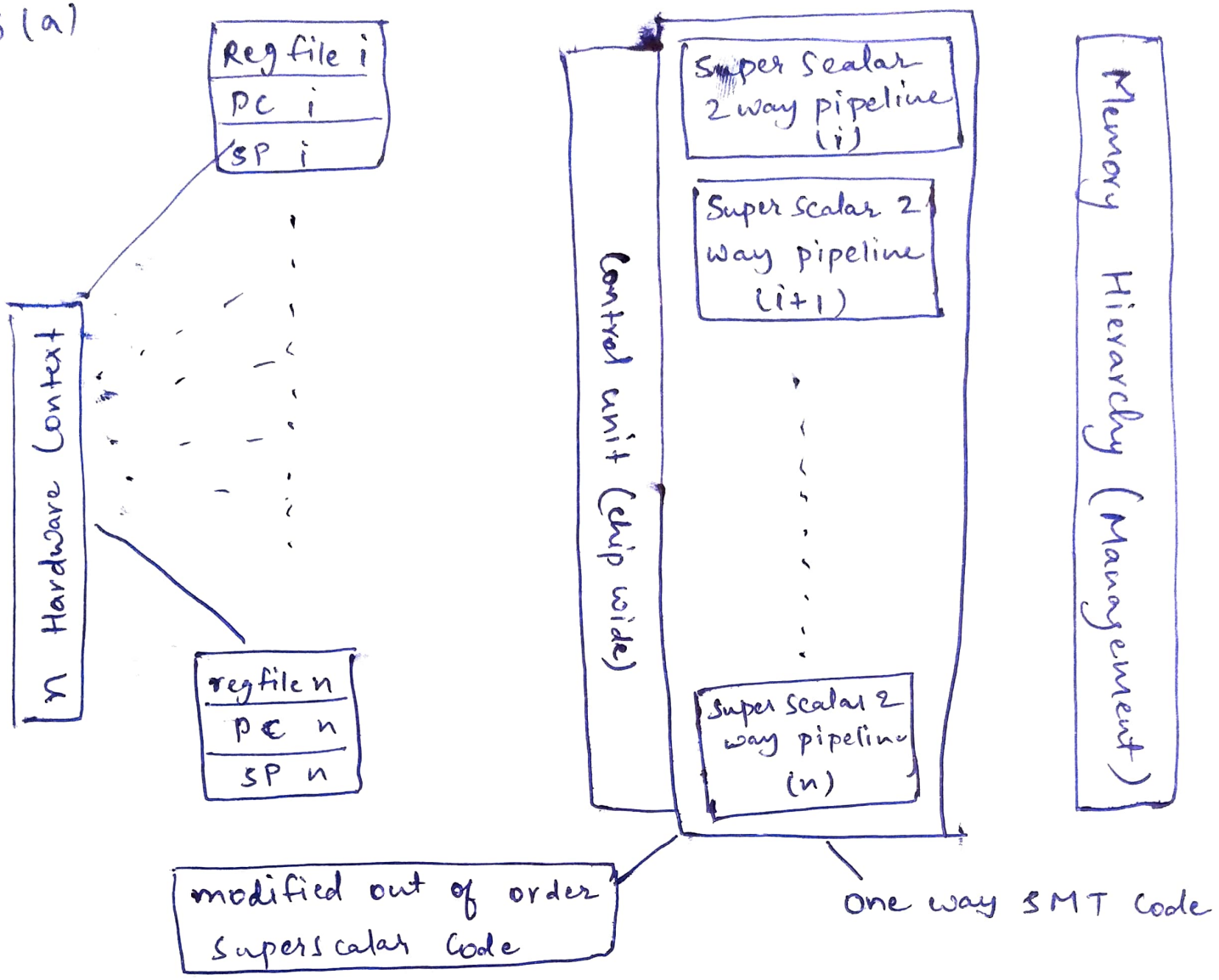
We also need a larger register file to be able to hold data from multiple threads, Also, for out of order execution, we need reservation stations and multiple functional units.

we assume 1 reservation station for 1 fus

Amar Kumar

Manoj Kumar

5 (a)




Amar Kumar

Manoj Kumar

5 (b) For multiple instruction issue and multiple Commit on top of it

→ we need to fetch multiple instruction at once

→ we will need to use dynamic branch prediction and speculation

→ VLIW processor

By dynamic issue

→ we need to issue multiple instruction in each clock cycle

→ we need to have more than one ALU and register files with additional ports so that we can avoid structural hazard

→ we generally extend to dynamic pipeline scheduling

→ we also need multiple buses to carry different data, to different write heads because we need to do multiple Commit.

5 (c) Multicore architecture with thread based execution

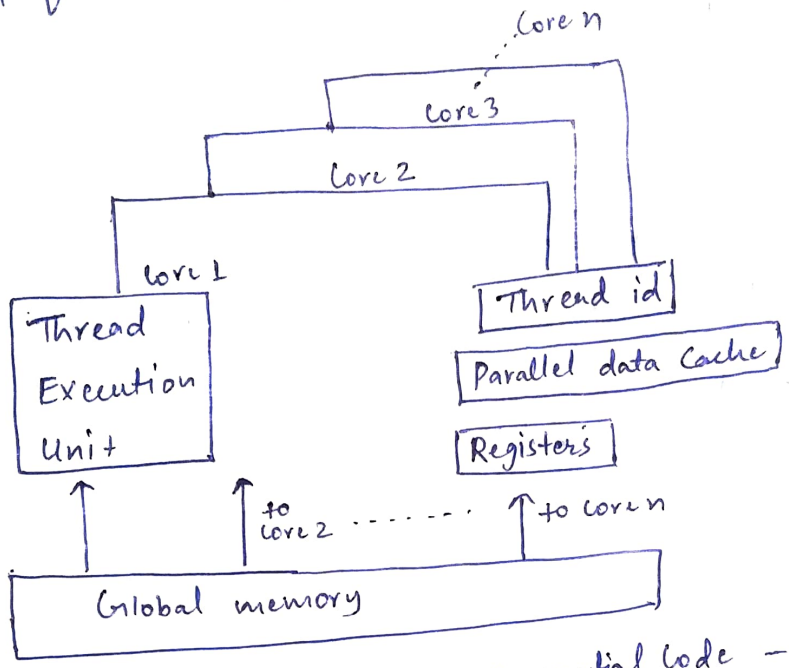
→ It will speedup the execution because even when we are using single core, we are using threads so that instructions can execute in parallel

→ When we use multiple cores on a single chip, it will provide upper hand in term of raw processing power

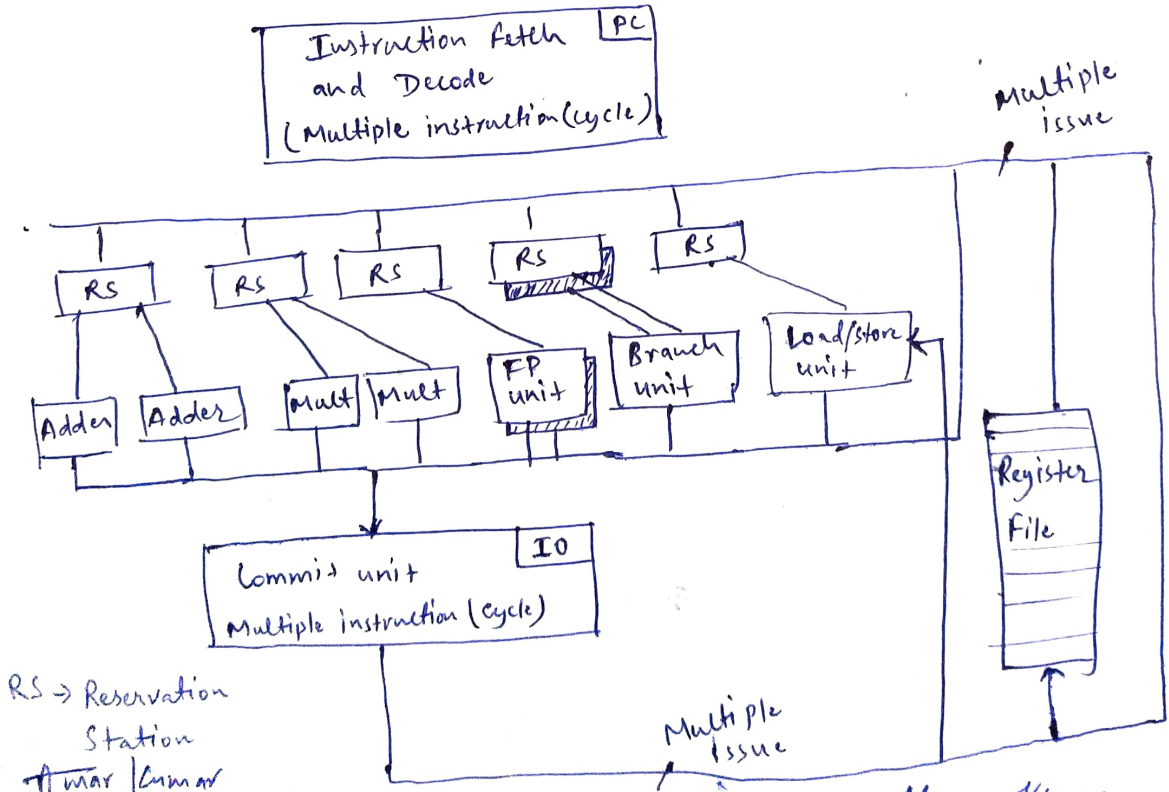
→ When we do true parallelism, we require multiple cores to be used in synchronised manner.

Amar Kumar Manoj Kumar

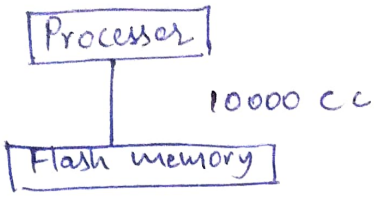
As illustrated, a process can be fragmented into smaller, more easily operable modules
Each of these modules can be scheduled to execute on different cores. After the desired output is achieved they can be recombined. hence enhancing the performance by completing the tasks simultaneously




Speed up = Execution time of sequential code
Execution time of threaded code

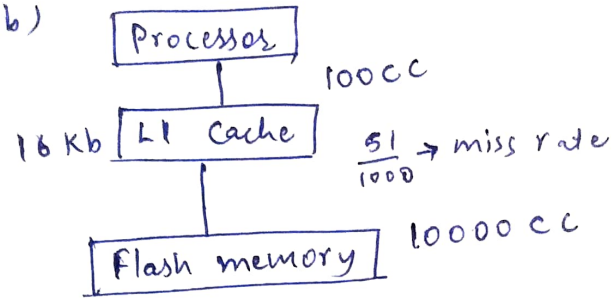


2(a)




AMAT = 10000 cc

(b)



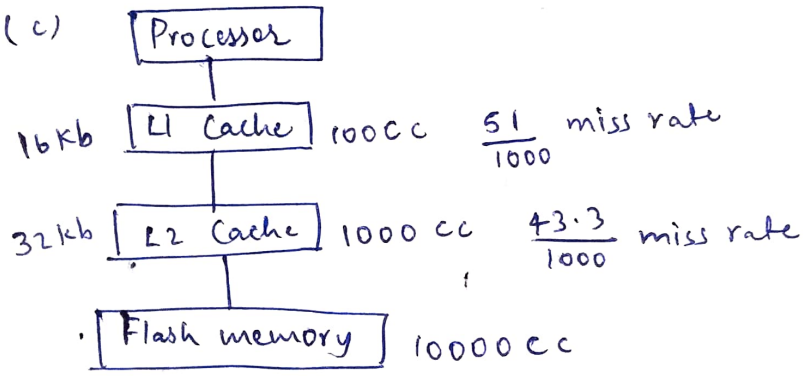

AMAT = Hit rate L1 + miss rate L1 * Miss penalty L1

$= 100 + \frac{51}{1000} \times 10000$

$= 100 + 510$

$= 610$ cc

(c)




AMAT = Hit rate L1 + Miss rate L1 * (Hit time L2 + Miss rate L2 * Miss penalty L2)

$= 100 + \frac{51}{1000} \left(1000 + \frac{43.3}{1000} \times 10000\right)$

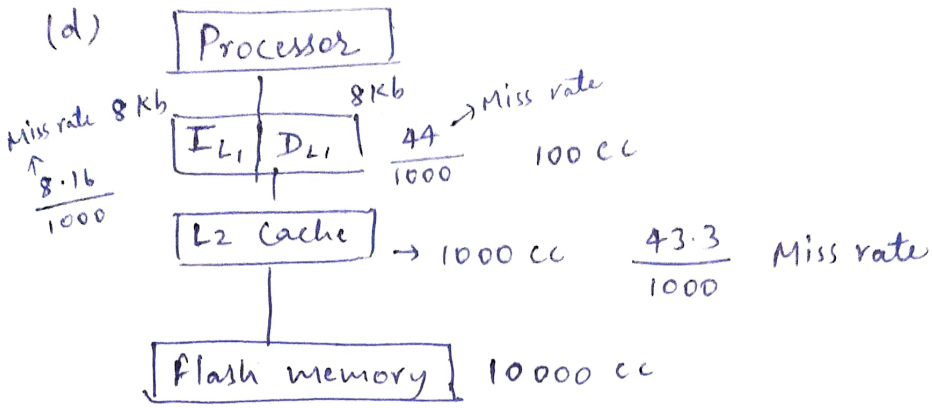
$= 100 + \frac{51}{1000} * 5330$

$= 371.83$ cc

Kumar Kumar

Manoj Kumar

(d)




75 % instruction, 25 % data

$$\text{AMAT} = \frac{75}{100}\left(100 + \frac{8.16}{1000}\left(1000 + \frac{43.3}{1000} \times 10000\right)\right) +$$

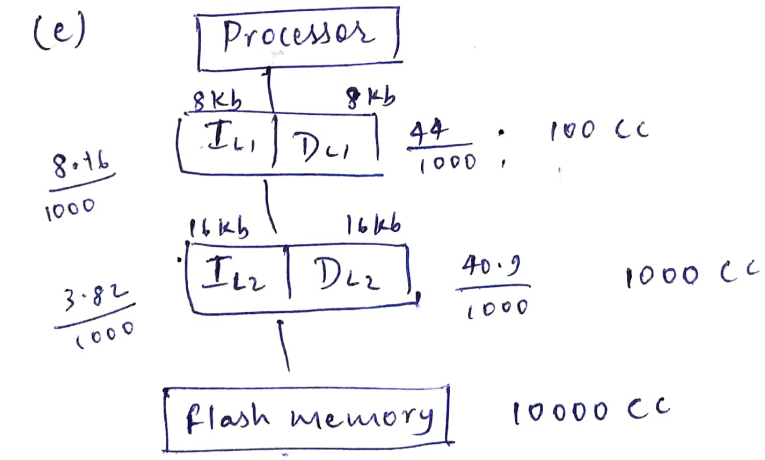
$$\frac{25}{100}\left(100 + \frac{44}{1000}\left(1000 + \frac{43.3}{1000} \times 10000\right)\right)$$

$$= \frac{3}{4}\left(100 + \frac{8.16}{1000}(1000 + 433)\right) + \frac{1}{4}\left(100 + \frac{44}{1000}(1000 + 433)\right)$$

$$= \frac{3}{4}\left(100 + \frac{8.16}{1000} * 1433\right) + \frac{1}{4}\left(100 + \frac{44}{1000} * 1433\right)$$

$$\approx 124 \text{ cc}$$

(e)




75 % instruction 25 % Data

Amar Kumar

Manoj Kumar

AMAT =

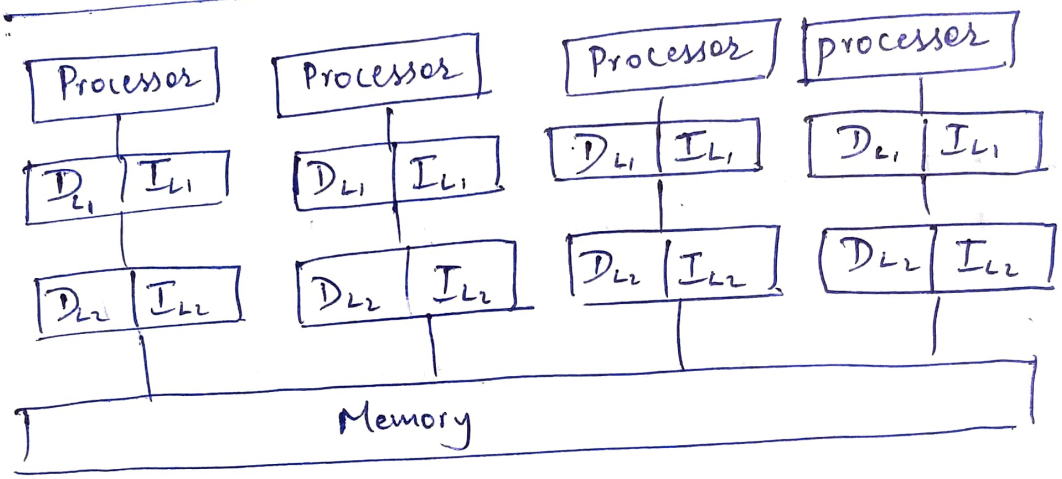
$$\frac{75}{100}\left(100 + \frac{8.16}{1000}\left(1000 + \frac{3.82}{1000} \times 10000\right)\right) +$$

$$\frac{25}{100}\left(100 + \frac{44}{1000}\left(1000 + \frac{40.9}{1000} \times 10000\right)\right)$$

$$= \frac{3}{4}\left(100 + \frac{8.16}{1000} * 1038.2\right) + \frac{1}{4}\left(100 + \frac{44}{1000} * 1409\right)$$

$$\approx 121.854 \text{ cc}$$

Better memory Hierarchy




AMAT =

$$\frac{1}{4}\left(\frac{75}{100}\left(100 + \frac{8.16}{1000}\left(1000 + \frac{3.82}{1000} \times 10000\right)\right) +\right.$$

$$\left.\frac{25}{100}\left(100 + \frac{44}{1000}\left(1000 + \frac{40.9}{1000} \times 10000\right)\right)\right)$$

$$= \frac{1}{4} * 121.854 \text{ cc}$$

$$= 30.4635 \text{ cc}$$

In this basically we are executing 4 units of (c) with single memory parallely to get faster access of Memory

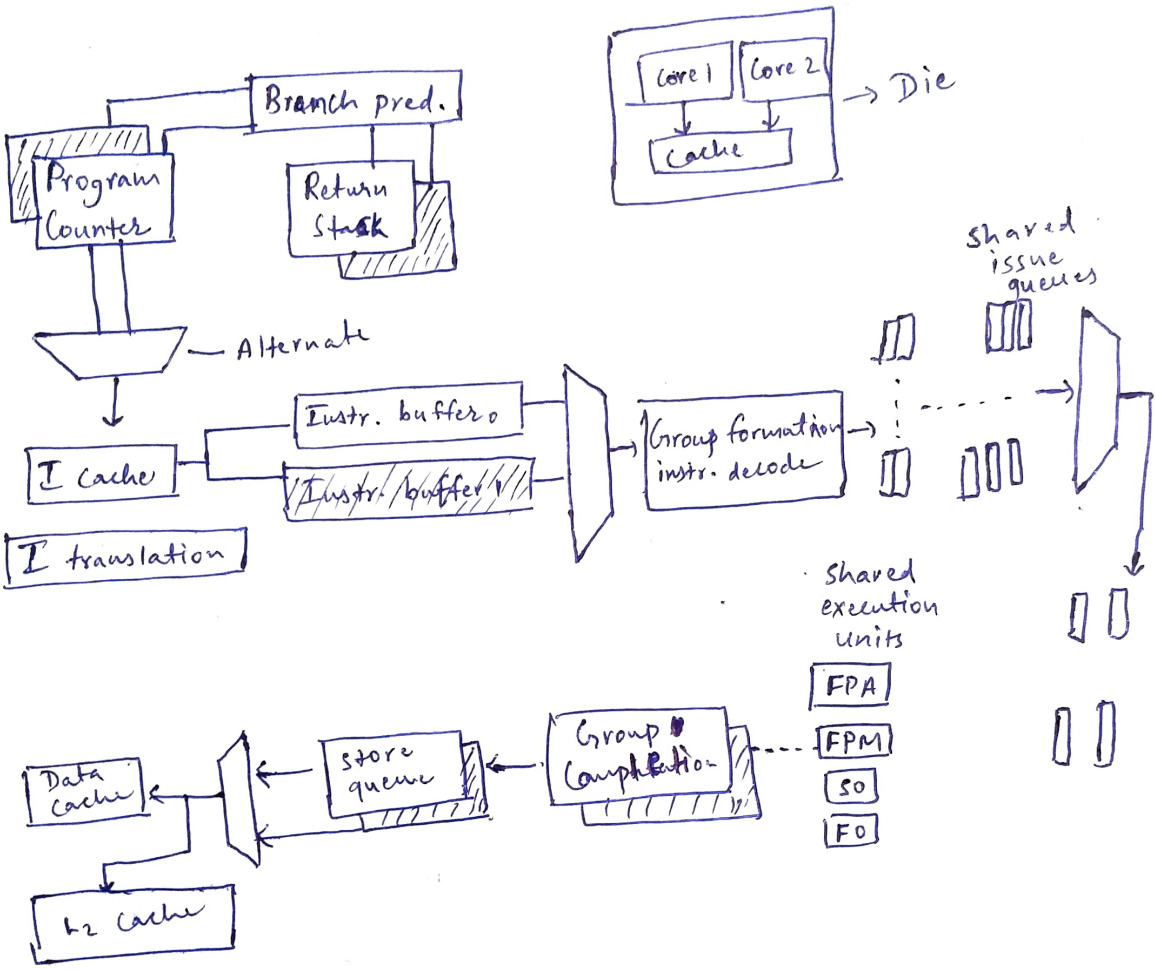
Amar Kumar

Manoj Kumar

1 (a) Space craft processor Chip

Needs high through put

- Can have multicore (2 core), may use only one based on load
- Each core has simultaneous multithreading support




(b) Thus the core SMT follows an approach similar to Power 5 processor

we can also do multicore systems to support multithreading in each core

So each core can have 2 simultaneous Multithreads running and we have 2 cores so this should

Amar Kumar

Manoj Kumar

give us enough lead soon for throughput. The two cores can be turned off at low load to conserve power while maintaining simultaneous multithreading. Hardware is required to support multicore, compared to having 2 processors each with SMT

(c) With SMT and 2 cores, we should be able to sum 4 different instructions on the chip (best case) while in superscalar we will only be able to run 1 instruction.

In a fully parallelised load, it should offer a little less than a line improvement compared to superscalar However we need to take the case

In each of SMT core two threads must not use/cause structural hazards else benefit of such instructions will be less.

If the load is not fully parallel, due to Ahamdal's law we see diminished returns.

Superscalar processor will cost less to manufacture as some addition hardware is used for multicore + to enable support for SMT.

Superscalar processor will although execute a single instruction more quicker one approach will have more throughput which was as desired

Amar Kumar

Manoj Kumar

(d) Given chip has to handle IO and instr. etc. simultaneously. obviously we can do much better with multicore + SMT since it can run multiple threads/instructions simultaneously.

(e) we can use flash memory to store Os and applications this can be EEPROMs or ROM which is much more robust than magnetic media and doesn't require mechinial parts thus improving longelisity

(f) we also would have on board DRAM to hold programs in execution. EEPROM & ROM to hold applications & OS as non volatile memory

(g) Our processor can have 2 cores each with $L_1$, $L_2$ private caches and $L_3$ shared cache and main memory last is ROM, EEPROM

3. (a) TYPE-B will have higher speed because it uses VLIW architecture in which instructions are bundled together into one instruction and executed parallely wherever TYPE-A has hardware dependent ILP and VLIW is compiler oriented

Amar Kumar

Manoj Kumar

(b) TYPE-A limitation

- Its hardware is complex
- Its ILP depends on hardware
- It is expensive

TYPE-B limitation

- Its compiler is complex
- Program size is large (explicit NOPs)
- VLIW memory system is complex
- parallelism is underutilized for few instruction set.

(c) Multiple thread support

- This will make overall execution of an instruction faster
- If a thread consist of cache misses, the other threads can continue, taking advantage of unused computing resources
- The architecture become complex, multiple threads can interfere with each other when sharing hardware resources

(d) uses of TYPE-A processer

- General computers like, AMD Athalon 64 process can do
- This include daily tasks involving FP math and arithmatic & Integer arithmatic in day to day tasks

Amar Kumar

Manoj Kumar

use of TYPE-B Processors

- Suitable for Digital Signal processing as we need to work with more than one number of images
- processing media data like Compression/decomposition of image & speed data

(c) Multiple Core helps in running multiple threads in parallel, overall speed of execution is increased, also since all functional unit are in parallel, this will help in multiple issue and out of order execution

TYPE-A

Processor limited by number of functional units, no. of buses, register access ports, large window size is impractical

TYPE-B

- statically finding parallelism. Code size
- No hazard detection in hardware
- Binary code compatibility

4. (a) Register renaming can be done to overcome register spill problem which is as follow

```
add  r0, r1, r2          add  A, r1, r2
mul  r0, r0, r3   renamed to  mul  B, A, t3
sw   r0, 0($sp)          sw   B, 0($sp)
add  r0, $0, $1          add  r0, $0, $1
```

This is done to avoid hazard caused by write Read & only read is hazard free on its own.

Amar Kumar

Manoj Kumar

(b) To resolve the register renaming issue in the hardware, we can do tomasulo's algorithm as inherently renaming a register by internally buffering to the reservation station. So we will have several extra registers each come specify to a reservation stations

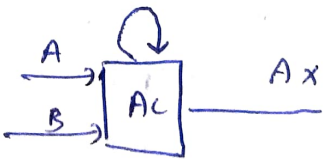
(c) The performance difference is that processor A is optimized for single threaded workload due to which it sometimes loses 60 % of its time to FP load, FP add etc. operations, whereas 4 core processor supports multithreading and thus can run 4 threads parallely.
We need high independent instructions that say could be executed in a fine grain multithreading by the processor to yield desired results.

(d) The change which we can do in architecture to make integer multiplier from integer adder is by doing add accumulates the number as many times as the smaller number. This could, potentially have its own clock which is much faster than a pipeline

Eg 2 X 3

P = 2 + 2

P = 2 + P




Amar Kumar

Manoj Kumar