Memory Access = MA

K200353 Mohsin Ali Mirza

① Implicit Parallelism

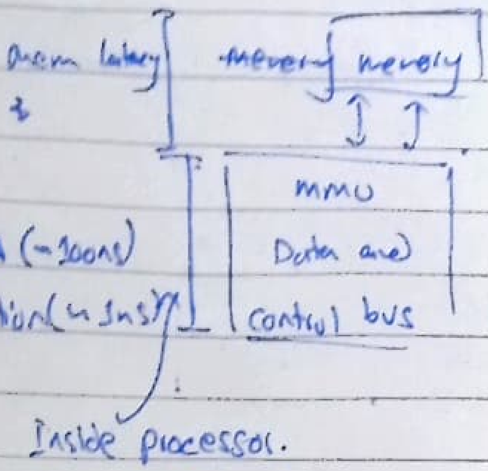
Instructions {

RD, R1, [2000H]

RD R2, [3000H]

Add R3, R1, 10 → Memory MA (~100ns)

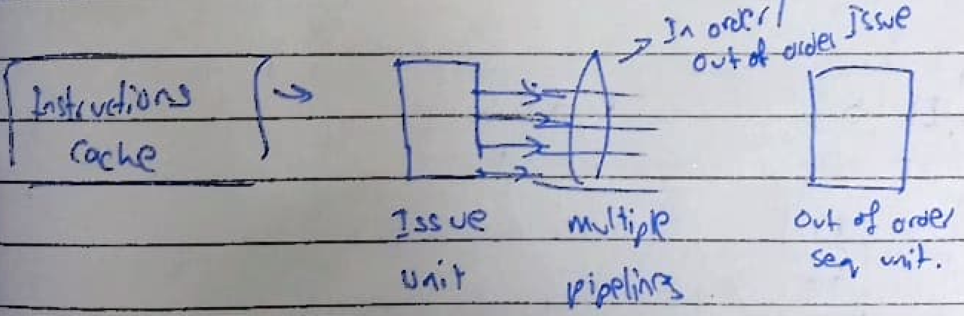
Add R3, R2, R3 → ALU Execution (~1ns)



② Pipelining (

IF	OF	OD	EXE	WR	
	IF	OF	OD	EXE	WR

③ SuperScalar Execution



④ VLIW (very Large Instructions word).

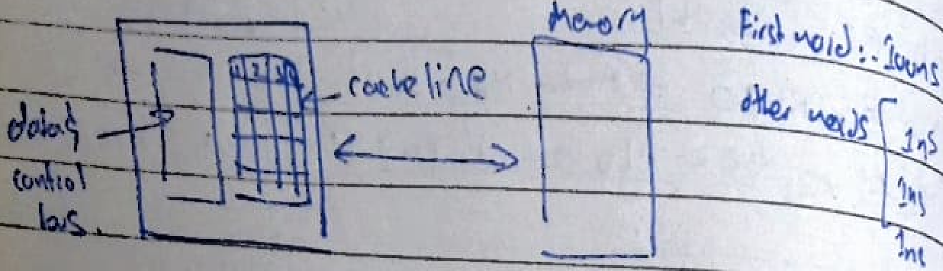
Ins1	Ins2	Ins3	. . . .

→ multiple instructions

↳ Block of instructions

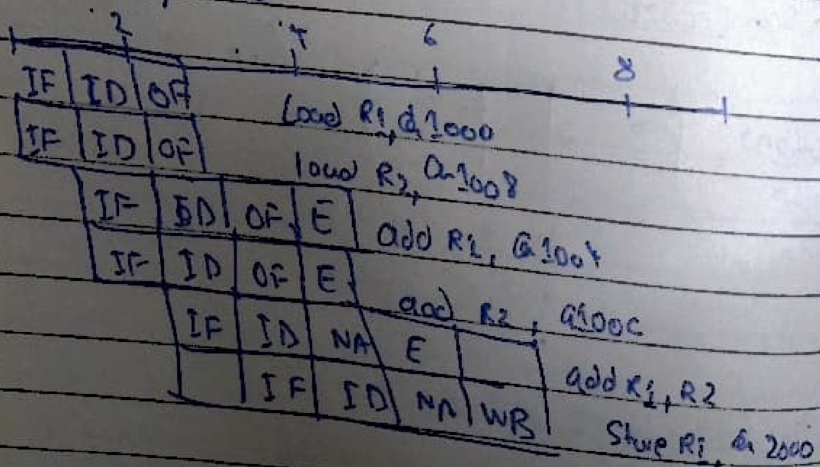
5 Cache Line

↳ Prefetching . four data is fetched and loaded in Cache In 1 access .

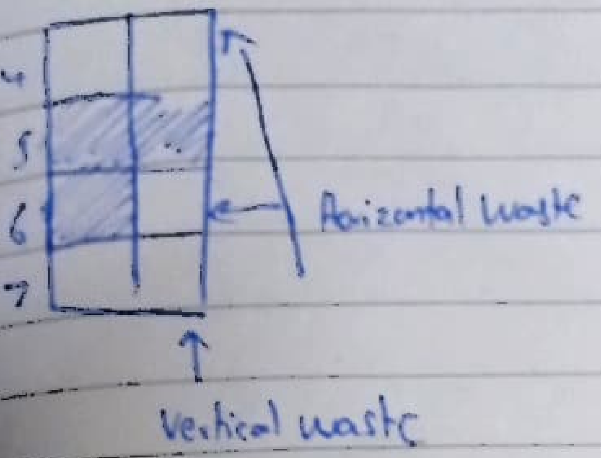


Execution Schedule for code fragments for superscalar pipe

1. load R1, @1000
2. load R2, @1008
3. add R1, @1004
4. add R2, @100C
5. add R1, R2
6. Store R1, @2000



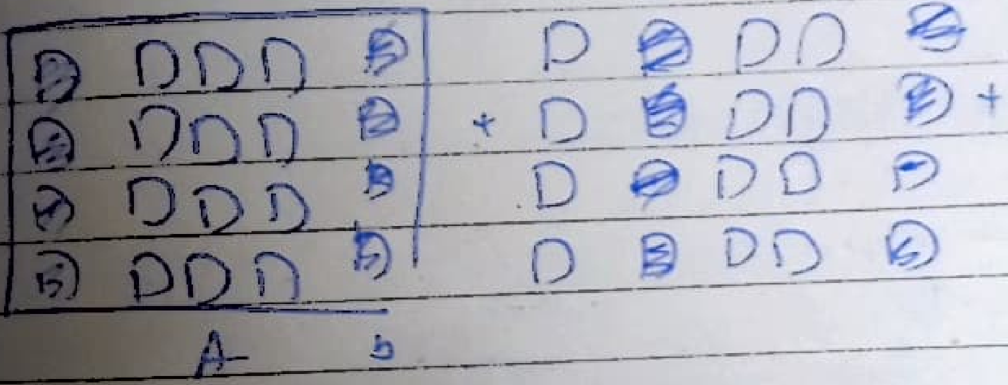
Hardware Utilization trace for schedule

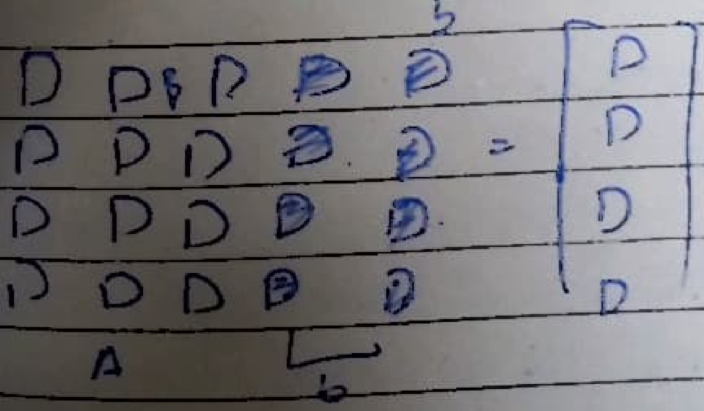


▨ Full issue slots

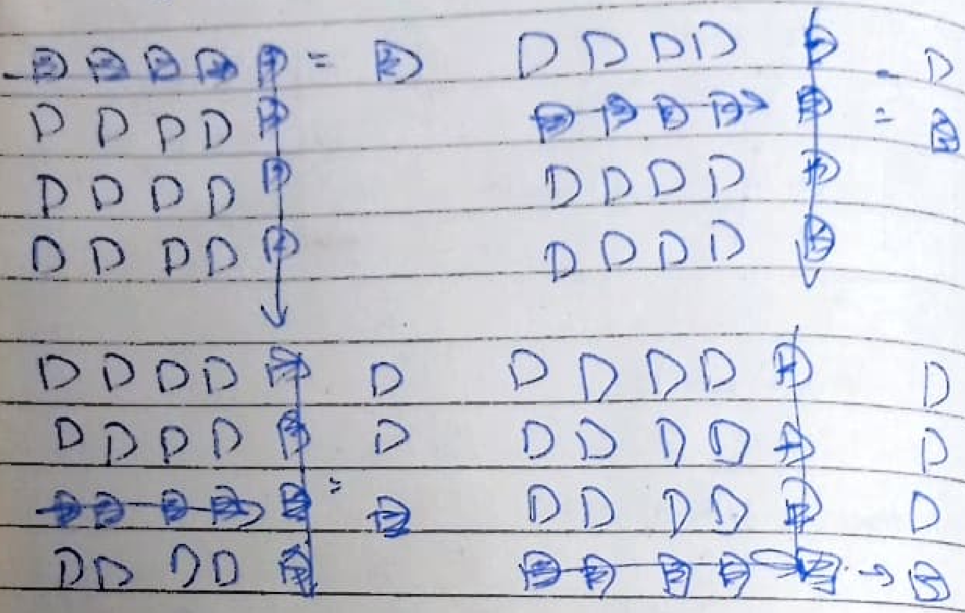
□ Empty issue slots

Column majority data access

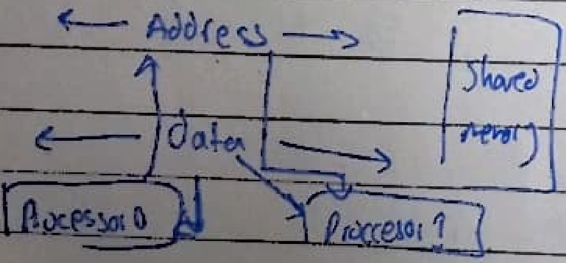


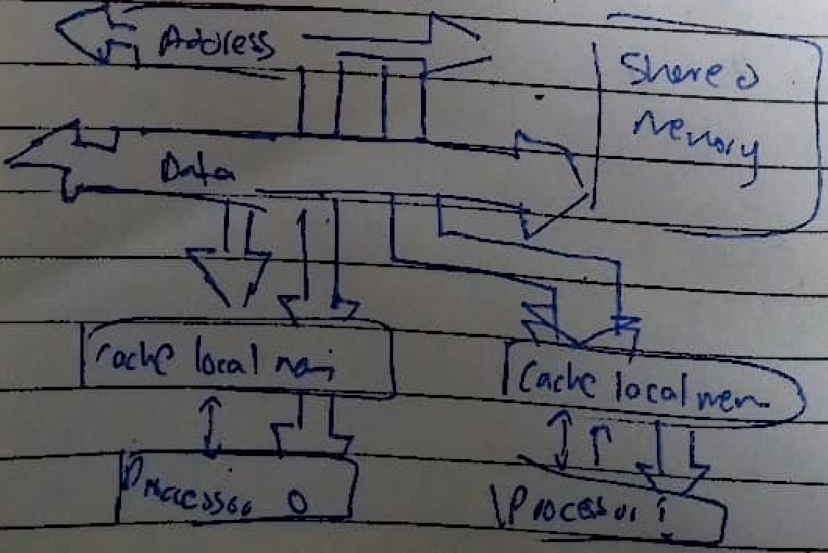


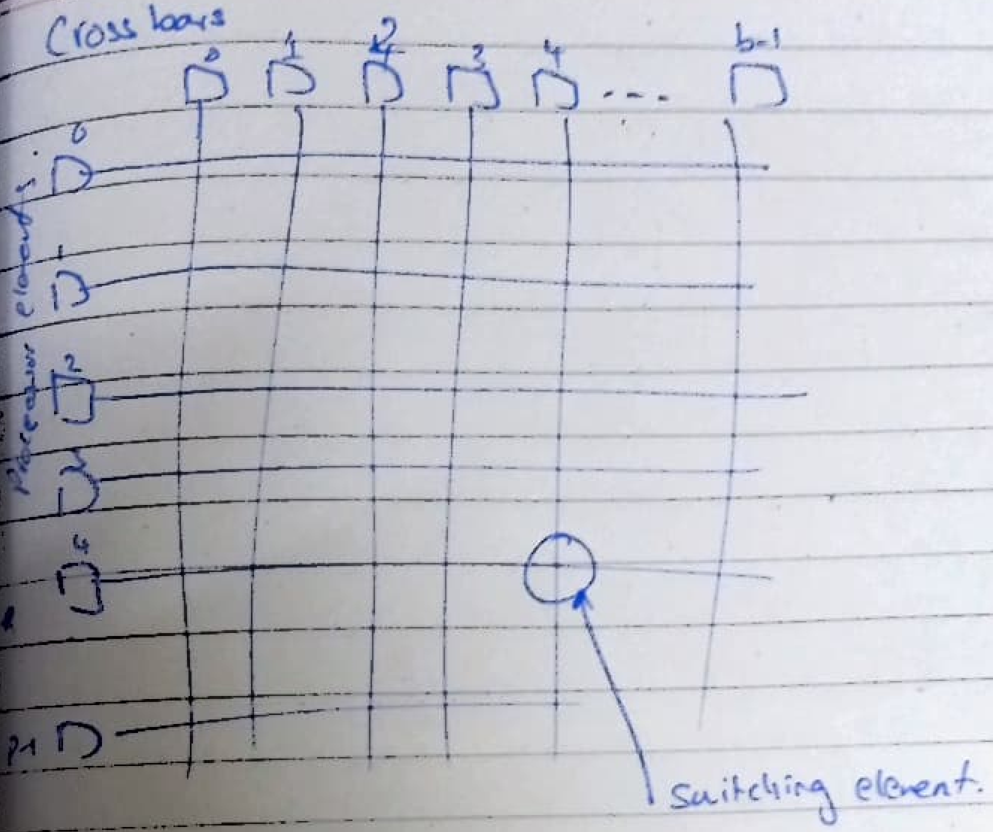
Row majority data acces :



Network Topologies : Buses







Ex 2.2 Effect of memory latency on performance

consider a processor operating at 1 GHz (1ns clock) connected to a DRAM with a latency of 100ns (no caches). Assume that processor has 2 mul add units and is capable of executing 4 instructions in each cycle of 1ns. The peak processor = $\frac{4}{1ns}$ = 4GFlops. Since memory latency is equal to 2 cycles and block size is 1 word, everytime a memory request is made, the processor must wait 100 cycles before it can process the data dot product of two vector element. It is easy to see the data of computation is limited to 1 flo FlOP every 100ns ~~rdin~~ or speed of 10 MFlops. A very small fraction of the peak processor rating. The example highlights the need for ~~ems~~ effective memory system performance in achieving high computation rate.

Ex 2.3 Impact of cache on memory system performance consider 1GHz processor with 100ns latency DRAM. In this case we intorduce a cache of size 32kb with a latency of 1ns ~~or~~ we use this setup to multiply ~~to~~ 2 matrices A & B, of dimensions 32x32. ~~we have~~ once again we assume an ideal cache placement strategy in which none of data items are over written by others fetching the two matrices into the cache correspon to fetching 2k words which takes approx 200µs. Multiplying n*n takes $2n^3$ operations which can be performed at 16k cycle. The total computation is ~~20~~ 216µs and peak processor computation of 64/216 = 303 MFlops.

Ex 2.4 Dot product of 2 vectors a memory system with a single cycle cache 100 cycle latency DRAM with the processor operating at 1GHz If the block size is ~~of~~ 1 word, the processor takes 100 cycle to fetch each word. The dot product performs are mul-add ie 2 Flops therefore the algo performs one flops every 100 cycles for a peak speed of of 10 MFlops as estimated. Now let us consider what happens if the block size is increased to 400 words i.e a processor can fetch a 4-word cache i.e every 100 cycle, Assuming that vectors are laid out linearly in memory: 4 mul-adds can be performed in 0 cycles. This is because a single memory access fetches ~~[illegible]~~ 4 elements of vector, This corresponds to a flop every 25ns