2102-COL216 Major Exam

Viraj Agashe

TOTAL POINTS

24 / 38

QUESTION 1

1 Processor Comparison 8 / 8

✓ **+ 2 pts** **Code for P1**

✓ **+ 2 pts** **Code for P2**

✓ **+ 1 pts** **Code for P3**

✓ **+ 1 pts** **Bytes transferred for P1**

✓ **+ 1 pts** **Bytes transferred for P2**

✓ **+ 1 pts** **Bytes transferred for P3**

+ 0 pts Incorrect or not attempted.

QUESTION 2

2 Adding ICB instruction 0 / 8

✓ **+ 0 pts** **incorrect/not attempted**

+ 2 pts Breaking up the code into clock cycles/steps

+ 1 pts Identifying new control states

+ 1 pts Describe control state transition

+ 2 pts Clocked assignments

+ 2 pts Glue logic

QUESTION 3

3 Hazards with Single Memory 6 / 7

+ 0 pts Incorrect/Incomplete/Unattempted

✓ **+ 2 pts** **Structural Hazard**

✓ **+ 2 pts** **Stall fetch and PC for 1 cycle**

✓ **+ 1 pts** **Number of stalls = nunber of mem instructions**

✓ **+ 2 pts** **CPI = 1 + (LS/N)**

- 1 Point adjustment

-1: No data hazards

1 -1: No data hazard

QUESTION 4

Caches and TLB 7 pts

4.1 Cache Inclusivity 2 / 3

✓ **+ 1 pts** **Inclusive property**

+ 0.5 pts Partial Correctness: Inclusive

+ 2 pts Exclusive Property

✓ **+ 1 pts** **Partial correctness: Exclusive**

+ 1 pts Only definition of inclusiveness and exclusiveness is given

+ 0 pts Not attempted/Incorrect answer

Every Liine is read from L1, therefore, line hit at L2 is sent to L1 and then displaced line is added to L2 and the sent line is removed from L2

4.2 2-Level Page Table 3 / 4

+ 0 pts Unattempted

+ 0 pts Wrong

+ 2 pts TLB entry structure complete (2 types of entry)

✓ **+ 1 pts** **TLB entry structure partial (1 type of entry)**

✓ **+ 2 pts** **Outcome complete (2 types of hits)**

+ 1 pts Outcome partial (1 type of hit)

QUESTION 5

Bus Arbitration and Throughput 8 pts

5.1 Bus Arbitration 3 / 4

+ 4 pts Correct

✓ **+ 2 pts** **Daisy chain-correct**

+ 1 pts Daisy chain-Arbitration mechanism

+ 2 pts NuBus,-correct

+ 1 pts NuBus,-Arbitration mechanism

+ 0 pts Incorrect

+ 0 pts Unattempted

+ 0 pts Click here to replace this description.

+ 1 Point adjustment

2 incorrect.....

5.2 Bus Throughput **2 / 4**

+ **0 pts** Incorrect/Unattempted

✓ **+ 2 pts without Split transaction**

+ **2 pts** With Split transaction

Name: VIRAJ AGASHE	Entry No.: 2020CS10567

COL 216 Computer Architecture : Major Test

Date: 08.04.2022 Time: 14:15-16:15 Max marks: 30

***Note**: Do any one out of questions 1 and 2. Do all of questions 3 to 5.*

1. Compare computation of statement s = (a + b) – (c + d) in three processors P1, P2 and P3 defined below in terms of number of instruction bytes and data bytes transferred between CPU and memory. Here a, b, c, d and s refer to memory locations. The expression is to be evaluated as it is, without any algebraic transformation or re-ordering of operations, that is, left addition followed by right addition followed by subtraction.

Proc	Processor type	Instruction size, data size	Relevant Instructions	
P1	accumulator based, 1-address	32-bit, 32-bit	load x	ACC ← Mem[x]
			store x	Mem[x] ← ACC
			add x	ACC ← ACC + Mem[x]
			sub x	ACC ← ACC – Mem[x]
P2	M-M type, 3-address	64-bit, 32-bit	add x, y, z	Mem[x] ← Mem[y] + Mem[z]
			sub x, y, z	Mem[x] ← Mem[y] – Mem[z]
P3	R-R type, 3-address, with registers r0 ... r15	32-bit, 32-bit	load Rd, x	Rd ← Mem[x]
			store Rd, x	Mem[x] ← Rd
			add Rd, Rn, Rm	Rd ← Rn + Rm
			sub Rd, Rn, Rm	Rd ← Rn – Rm
			Rd, Rn, Rm ∈ {r0, r1, . . . , r15}	

(8)

Sol. **P1:** Instructions

- load a ✓
- add b ✓
- store a ✓
- load c ✓
- add d ✓
- store c ✓
- load a ✓
- sub c ✓
- store s ✓

✓ → Data transfer from proc. to mem. or vice versa

No. of instruction bytes

= (No. of instructions) × (Instruction size)

= 9 × 4 bytes = 36 bytes

No. of data bytes transferred

= (No. Instructions containing data transfer) × (4 bytes)

= 9 × 4 = 36 bytes

=

P2: Instructions

- add s_1, a, b
- add s_2, c, d
- sub s, s_1, s_2

No. of instruction bytes ⇒ 8 bytes

= (No. of instructions) × (Instruction size)

= 3 × 8 = 24 bytes

No. of data bytes transferred

add x, y, z → Load y, z from memory into ALU, process & load back result.

Similiarly with sub x, y, z

∴ Each instruction involves 3 data transfer from processor to memory or vice-versa.

∴ No. of data bytes transferred

= 3 × 3 × 4 = 36 bytes

P3. Instructions

> load R1, a
> load R2, b
> add R3, R1, R2
> load R4, c
> load R5, d
> add R6, R4, R5
> sub R7, R3, R6
> store R7, s

No. of instruction bytes

= (No. of instructions) x (~~No. of~~ Instruction size)

= 8 x (4 bytes)

= 32 bytes

No. of data bytes transferred

Data is transferred from proc. to mem. once in each load and once in each store.

∴ Data transferred = (4x1 + 1x1) x 4 bytes

= 20 bytes

Conclusion: The no. of instruction bytes is least in P2 (as we are able to represent in very few instructions).

The no. of data bytes transferred from the memory to processor & vice-versa is least in P3 (as we don't need to store in memory each time, we can store in the local reg files)

The total no. of bytes transferred from memory to CPU is minimum in P3 (52 bytes). Thus from memory access perspective, P3 is the most efficient.

Name: VIRAJ AGASHE	Entry No.: 2020CS10537

COL 216 Computer Architecture : Major Test

Date: 08.04.2022 Time: 14:15-16:15 Max marks: 30

2. Consider a multi-cycle processor design (like assignment 2, stage 3) executing ARM instructions {add, sub, cmp, mov, ldr, str, beq, bne, b} with limited variants. A new instruction icb (increment, compare and branch), as defined below, is to be added to this processor to facilitate implementation of loops.

icb Rd, Rn, Rm Rd ← Rd + 1; if Rd < Rn then PC ← Rm else PC ← PC + 4

What new control states would you introduce and what actions would be performed in those states? What changes/additions will be required in the glue logic? Whole design is not to be given - just write small fragments of VHDL code to make your answer about changes/additions precise. Make no change in register file (continue with 2 read and 1 write ports). Use ALU for all additions and comparisons. Assume that "1100" in bits 27-24 of instruction identify icb. Use any three 4-bit fields for specifying 3 registers.

(8)

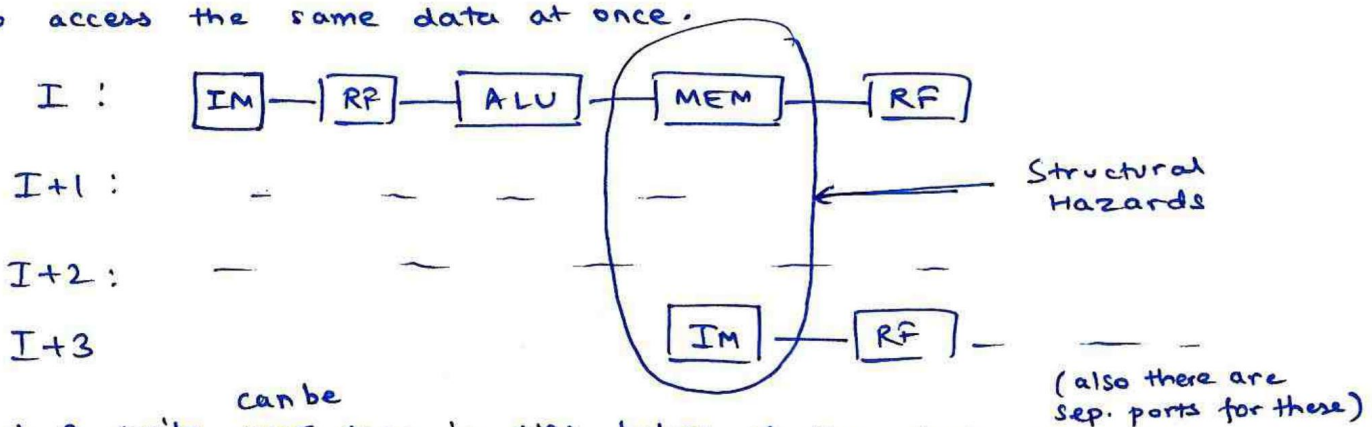
Name: VIRAJ AGASHE	Entry No.: 2020CS10567

COL 216 Computer Architecture : Major Test

Date: 08.04.2022 Time: 14:15-16:15 Max marks: 30

3. Suppose we have a 5-stage pipelined processor with stages - IF (instruction fetch), ID (instruction decode/operand read), EX (execute), Mem (memory access) and WB (write back). What kind of hazards will be introduced if we have a single memory rather than separate program and data memories? How would you dynamically check for these hazards and insert stall cycles (ignore other types of hazards)? What would the number of stall cycles (average) depend on? Find an expression for average CPI accounting for the stall cycles. (7)

Sol. If we introduce a single memory for data & instructions, we would introduce structural hazards, i.e. 2 instructions wanting to access the same data at once.



Read & write can be done in diff. halves of the clock cycle, so the hazard will occur when we are "reading" from both IM and DM. (as there is only one read port). (also there are sep. ports for these)

→ We can dynamically check for these errors / hazards. If we encounter an ldr instruction in the pipeline (i.e. reading from memory) then we know that the instruction I+3 in the pipeline will encounter a data hazard. Therefore we can introduce 1 stall cycle into the pipeline after the instruction I+2. Therefore, by the time the I+3 instruction wants to access the memory, the read (corresponding to ldr) would have been performed.

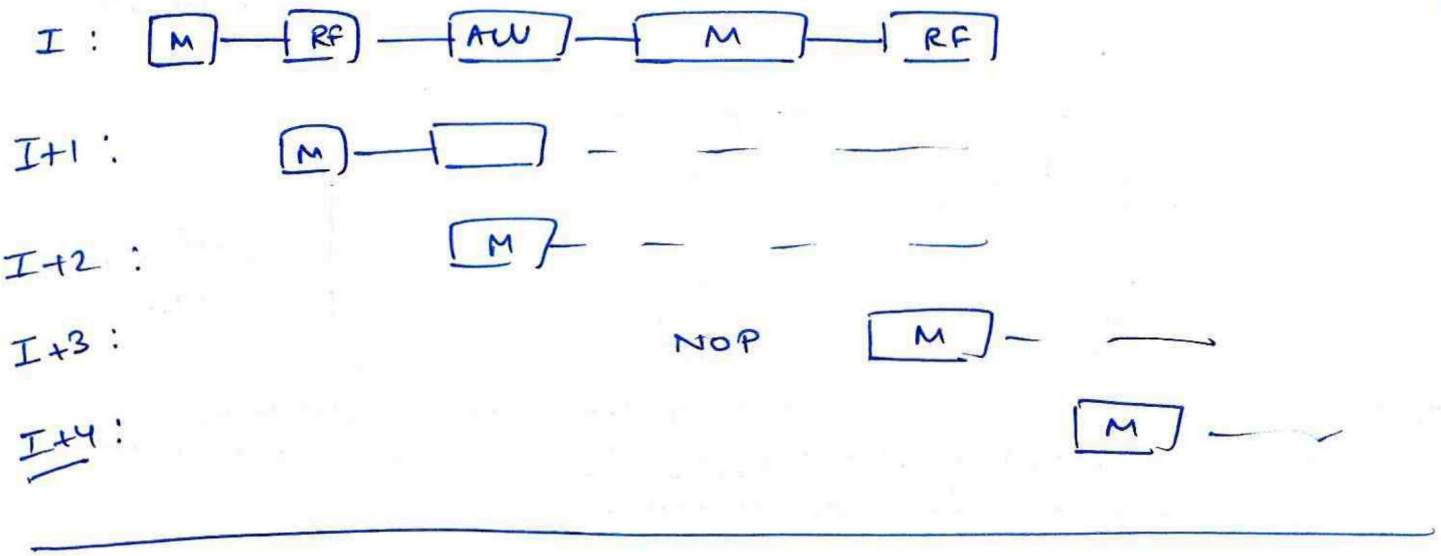
→ The no. of stall cycles would depend on the no. of instructions containing a load (ldr) instruction (fraction of instructions)

Let the fraction of ld- instructions be f. Then, we have,

$$CPI_{new} = \frac{f \times I \times 2 + (1-f)\, I \times 1}{1 \times I}$$

$$\boxed{CPI_{new} = 1 + f}$$

How stalling removes hazard



Name: VIRAJ AGASHE	Entry No.: 2020CS10567

COL 216 Computer Architecture : Major Test

Date: 08.04.2022 Time: 14:15-16:15 Max marks: 30

4. (a) In a system with two levels of cache, what actions would need to be taken to ensure inclusiveness? What actions would need to be taken to ensure exclusiveness?

(b) A fully associative TLB for a simple virtual memory has the following fields in each of its entries: valid bit, virtual page number and physical page number. How would you modify the TLB structure to support a virtual memory with two level page tables (that is, segmented virtual memory)? What would be various possible outcomes when this TLB is looked up? (7)

Q4. (b) We can modify the TLB for 2-level page table as follows: Along with the VPN, we would also need to store which segment of the

Q4. Inclusive cache: All data in present in L1 cache is also present in L2 cache.

Exclusive cache: Any data present in L1 cache is absent from the L2 cache.

Steps for inclusivity:

→ Whenever there is a write hit in the L1 cache, we would also need to write the data into the L2 cache.

→ Whenever there is a write miss in both L1 & L2, we should load the data from memory into both L1 and L2 (if we are following such a write policy) or either only into L2.

→ When there is a read-miss, again we should either write from memory into both L1 & L2 or into neither.

→ When evicting a block from L2 cache we should evict it from L1 cache as well.

Steps for exclusivity

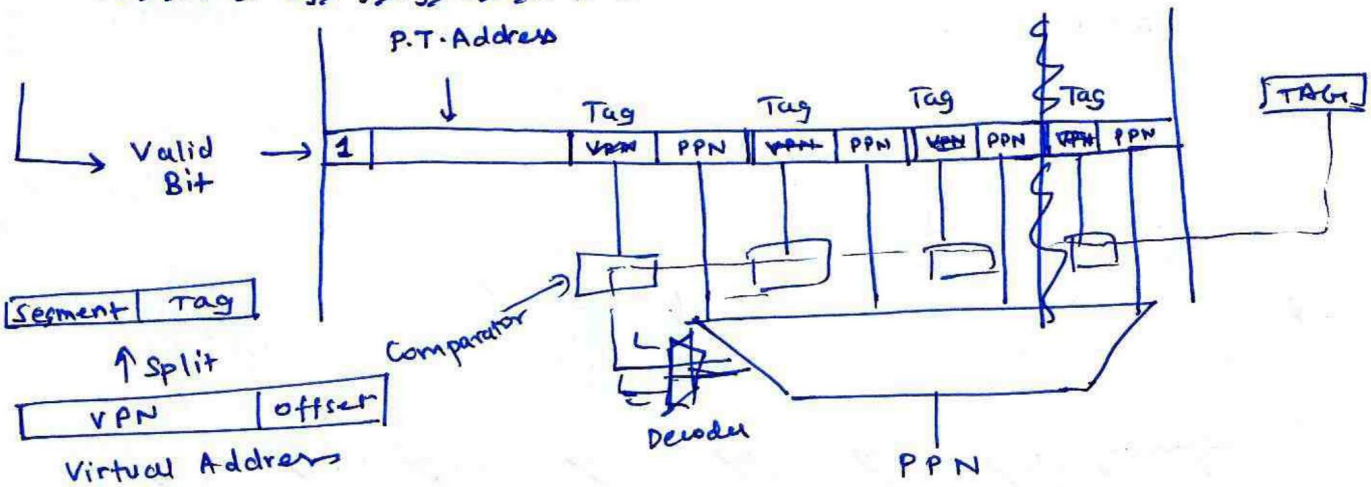
→ When there is a write miss in both L1 & L2 cache, load from the memory only into L1 or only into L2 but not both.

→ Similiarly for read miss, depending on the kind of policy followed we should either load into L1 or L2 but not both.

→ When we evict a block from L1 cache we can move it into L2 cache to maintain efficiency of the 2-level cache.

Q4. (b) To support a virtual memory with 2-level page tables, we can modify the TLB as follows:

~~* Instead of fully associative,~~

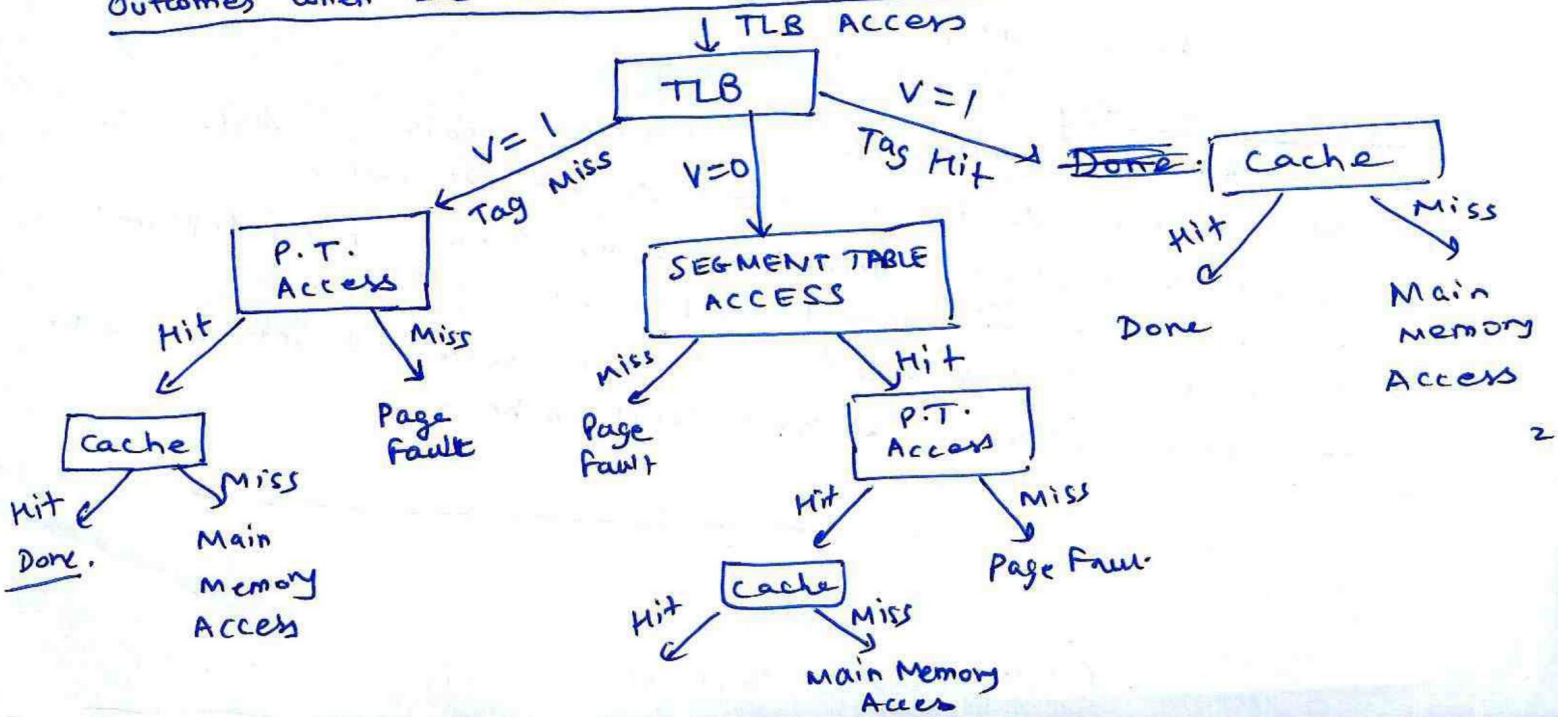


TLB for 2 level (segmented) structure

Explanation: We use the segment bits from the virtual address to index into the TLB. In the TLB we store a valid bit, address of the page table (corresponding to the segment bits if valid bit = 1) and 4-way associative ~~cache~~ table, i.e. the PPN corresponding to the VPN can be at 4 possible locations.

This helps us as, when we index into the TLB using the segment bits, even if the PPN corresponding to the VPN we want is not present, if V=1, we know that the P.T. address is valid & we don't need to access the segment table, instead can directly access the page table pointed to by the address. And, if we get a hit with the tag, we get the actual PPN as well.

Outcomes when TLB are accessed are best represented by a flowchart:



Name: VIRAJ AGASHE	Entry No.: 2020CS10587

COL 216 Computer Architecture : Major Test

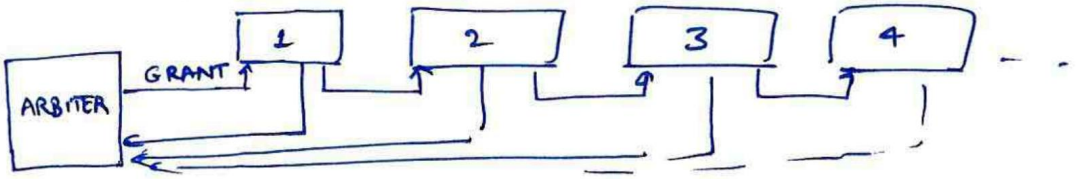
Date: 08.04.2022 Time: 14:15-16:15 Max marks: 30

5. (a) Consider two schemes for bus arbitration - (i) daisy chaining and (ii) distributed arbitration by self selection (like NuBus). In (i), the bus grant signal is chained through various masters in priority order. In (ii), IDs of the masters representing their priorities are put on the bus through some logic such that ID of the highest priority requester gets through. Suppose we want the arbitration to take place within one bus clock cycle. How would the cycle time depend on the number of masters in the two schemes? Give justification for your answer.

(b) A system with multiple disks is required to run programs that are dominated by disk I/O. Data transfer between a disk and memory takes place using DMA over a 32-bit wide synchronous bus with a 500 MHz clock in bursts of length 16. Each transfer requires 2 cycles to initiate (sending address etc), followed by a gap of 20 cycles (due to memory latency) and then 16 cycles for the data burst. (i) What is the peak throughput on the bus possible for such transfers? (ii) How will this figure change if split transactions are supported on the bus?

(8)

Q5. (a) (i) In daisy chaining, the grant signal is chained, i.e. a master will let the grant signal pass through it only if it has not put in a request at that time (rising edge).



In daisy chaining, the arbitration occurs as ~~can occur independent of the no of masters. This is because the grant signal is chained~~, whichever master is the highest priority will block the grant signal at the rising edge (if it has made a request). There is no separate logic involving the no. of masters (there may be some propogation delay due to the no. of masters and the distance of the last master from the arbiter, however this delay is not that significant). (compared to ii))

This delay would increase as O(n), where n is the no. of masters

(ii) In NuBus, the logic of the calculation of the ID of highest priority depends on the no. of masters. For example, in a 4 bit ID scheme

~~$ID_i = req_i \cdot ID_i (ID_{i-1} + arb'_{i-1})(ID_{i-2} + arb'_{i-1}) = -$~~

$ID'_2 = (req_2) \cdot ID_2 \cdot (ID_3 + arb'_3)$

$ID'_1 = (req_1)\, ID_1 \cdot (ID_3 + arb'_3)(ID_2 + arb'_2)$

!

Clearly this logic depends on the no. of masters as the computation of ID'_0 depends on n as O(n). i.e $ID'_0 \propto n$

~~Thus, the cycle time would vary linearly with the no. of masters in the NuBus scheme.~~

In fact, $ID_0 \propto n = kn$

$ID_1 \propto n-1 = k(n-1)$

$\vdots$

$ID_{n-1} \propto 1 = k(1)$

$\therefore$ Total time $= kn + k(n-1) + \ldots + k(1)$

$\propto O(n^2)$.

So the clock cycle in Nubus increases quadratically with the no. of masters, i.e. $\propto n^2$.

Qs.(b) (i) For peak throughput (let us assume address phase can be pipelined)
Note that we will need (20+16) cycles (since address phase is pipelined) per burst.

Amount of data transferred = 32 x 16 bits

Clock period $= \frac{1}{f} = \frac{1}{500 \times 10^6}$ s

$\therefore$ Peak throughput $= \frac{\text{Data Transferred}}{\text{Time}} = \frac{32 \times 16 \text{ bits}}{\text{Cycles} \times \text{Clock Period}}$

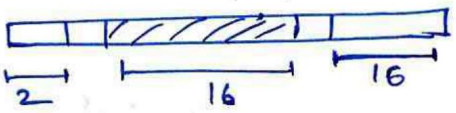
$= \frac{32 \times 16}{36 \times \frac{1}{f}} = \frac{32 \times 16 \times 500 \times 10^6}{36}$

$= \frac{640}{9} \times 10^8$

$\simeq 7.1 \times 10^9$ bits/sec

(Note that if we do not assume address pipelining then in above expression we will use no. of. cycles = 20 + 16 + 2 = 38)

(ii) If split transactions are permitted we can send the 16 burst of the previous address read during the memory latency time of 20 cycles. Now, the peak throughput will be, (still assuming pipelining).



→ we are able to transfer double the data in the same time

Peak throughput $= \frac{\text{Data transferred}}{\text{Time}} = \frac{2 \times 32 \times 16}{36 \times 1/f}$, {f = 500 MHz}

$=$ _____ Ans.

The figure (acc. to our assumptions) will be 2x the ans for part (i). (i.e. higher peak throughput is achieved)