University of California at Berkeley
College of Engineering
Computer Science Division - EECS

CS 152 D. Patterson & R. Yung
Fall 1995

Computer Architecture and Engineering
Midterm II

Your Name:	
SID Number:	
Discussion TA(s):	

You may use two pages of notes. You have 180 minutes.
Please write your name on this cover sheet and also at the top left of each page.
The point value of each question is indicated in brackets after it.
Please show your work. Write neatly and be well organized. Good luck!

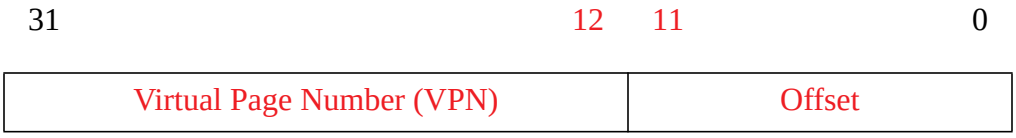
Problem	Score
1	/ 27
2	/ 13
3	/ 30
4	/ 20
Total	**/ 90**

Question 1: Virtual Memory

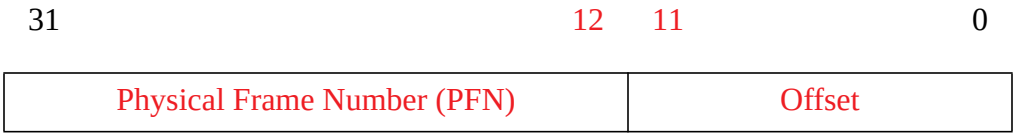
An eight-entry direct-mapped TLB is implemented in the current design. Both the virtual and physical addresses are 32 bits wide, and page size is 4kB. Address translation is performed by this TLB for every memory access. **NOTE:** All addresses are given in hexadecimal.

a) Label the virtual and physical address fields used in address translation. **[4 pt]**

Virtual Address



Physical Address



-1 for having VPN=[31:13], PFN=[12:0]
-2 for all other mistakes

b) If a valid translation is found and access protection is not violated, the corresponding physical frame number is used to generate the physical address.

If there is a TLB miss, the TLB is refilled with the next available physical frame number starting at 0x3. (You may assume that every TLB miss also causes a page fault.) The access protection for the new page should be read- and write-enabled. If access protection is violated, the access should be aborted and no physical frame number generated.

The initial state of the TLB is given in Table 1. For the four memory accesses in Table 2, show the translated physical addresses and indicate fault types, if any, based on the initial TLB state. **[16 pt]**

NOTE: the memory accesses should be done in the order shown.

Table 1: Initial TLB state (do not modify this table)

Index	Virtual Page Number	Physical Frame Number	Read	Write	Valid
0					0
1	0x1	0x0	1	0	1
2					0
3					0
4	0x104	0x1	1	1	1
5	0x4005	0x2	1	1	0
6					0
7					0

Table 2: Memory accesses

Virtual address	Physical address	Fault type, if any
0x1008 (read)	0x0008	none
0x0 (read)	0x3000	TLB miss
0x4005000 (read)	0x4000	TLB miss
0x10BC (write)		Write access viol.

Per line:
-1 for missing 0 or offset value
-2 for all other mistakes

c) Fill in the content of the TLB after the four memory accesses. **[7 pt]**

Table 3: Final TLB state

Index	Virtual Page Number	Physical Frame Number	Read	Write	Valid
0	0x0	0x3	1	1	1
1	0x1	0x0	1	0	1
2					0
3					0
4	0x104	0x1	1	1	1
5	0x4005	0x4	1	1	1
6					0
7					0

1 pt. for each correct row

Question 2: I/O

A 1993 3.5 inch IBM disk rotates at 4318 revolutions-per-minute (RPM), has a random seek time of 11ms, transfers at 4 MB/s, has a capacity of 1 GB, and the mean-time-to-failure (MTTF) is 400,000 hours. The SCSI controller overhead is 2ms.

A 1995 3.5 inch IBM disk rotates at 7200 RPM, has a random seek time of 8ms, transfers at 12 MB/s, has a capacity is 4.2 GB, and the MTTF is 1,000,000 hours. The SCSI controller overhead today is 1ms.

a) On average, how much faster is the new disk than the old disk for a read of 4 kB assuming random seeks? Assume the disks are idle so that there is no waiting time. **[4 pt]**

$$t = t_{SEEK} + \frac{1}{2}t_{ROT} + t_{OH} + \frac{xfersize}{BW}$$

$$t_{OLD} = 11ms + \frac{60\frac{\text{sec}}{min} \times 1000\frac{ms}{\text{sec}}}{2 \times 4318\frac{rev}{min}} + 2ms + \frac{4kB \times 1000\frac{ms}{\text{sec}}}{4\frac{MB}{\text{sec}} \times 1000\frac{kB}{MB}} = 20.95ms$$

$$t_{NEW} = 8ms + \frac{60\frac{\text{sec}}{min} \times 1000\frac{ms}{\text{sec}}}{2 \times 7200\frac{rev}{min}} + 1ms + \frac{4kB \times 1000\frac{ms}{\text{sec}}}{12\frac{MB}{\text{sec}} \times 1000\frac{kB}{MB}} = 13.50ms$$

$$Speedup = \frac{t_{OLD}}{t_{NEW}} = \frac{20.95ms}{13.50ms} = 1.55 times faster$$

b) If the actual seek time is 25% of the random seek, how much faster is the new disk now? **[4 pt]**

$$t_{OLD} = 0.25 \times 11ms + \frac{60\frac{\text{sec}}{min} \times 1000\frac{ms}{\text{sec}}}{2 \times 4318\frac{rev}{min}} + 2ms + \frac{4kB \times 1000\frac{ms}{\text{sec}}}{4\frac{MB}{\text{sec}} \times 1000\frac{kB}{MB}} = 12.70ms$$

$$t_{NEW} = 0.25 \times 8ms + \frac{60\frac{\text{sec}}{min} \times 1000\frac{ms}{\text{sec}}}{2 \times 7200\frac{rev}{min}} + 1ms + \frac{4kB \times 1000\frac{ms}{\text{sec}}}{12\frac{MB}{\text{sec}} \times 1000\frac{kB}{MB}} = 7.50ms$$

$$Speedup = \frac{12.70ms}{7.50ms} = 1.69 times faster$$

c) Now assume the read size is 1 megabyte, with seeks being 25% of random time. How much faster is the new disk? **[4 pt]**

$$t_{OLD} = 0.25 \times 11ms + \frac{60\frac{\text{sec}}{min} \times 1000\frac{ms}{\text{sec}}}{2 \times 4318\frac{rev}{min}} + 2ms + \frac{1MB \times 1000\frac{ms}{\text{sec}}}{4\frac{MB}{\text{sec}}} = 261.70ms$$

$$t_{NEW} = 0.25 \times 8ms + \frac{60\frac{\text{sec}}{min} \times 1000\frac{ms}{\text{sec}}}{2 \times 7200\frac{rev}{min}} + 1ms + \frac{1MB \times 1000\frac{ms}{\text{sec}}}{12\frac{MB}{\text{sec}}} = 90.50ms$$

$$Speedup = \frac{261.70ms}{90.50ms} = 2.89 times faster$$

d) What does this performance change in just two years suggested in the design of computer systems? **[1 pt]**

For **a**, **b**, **c**:
- Get wrong speedup consistently: -1
- Leave off adapter overhead: -3
- Leave off rotation time: -3
- Off by factor of 100: -2

Poss. ans. for **d**: Since disks have become 1.69 times faster for small transfers and 2.89 times faster for large transfers in the last two years, it suggests that larger disk transfers will be encouraged in the future (e.g., larger page sizes).

Question 3: Caches

In your first job out of school, you are given the task of evaluating the performance of four data cache designs (shown inTable 4). The best one will be incorporated in the next-generation processor design. The caches are accessed with 16-bit virtual addresses.

Table 4: Four cache configurations

	Capacity	Associativity	Block Size	Sub-block Size	Write Policy
A	32B	direct-mapped	8B	4B	no write allocate
B	32B	2-way	8B	none	no write allocate
C	32B	direct-mapped	8B	none	write allocate
D	32B	2-way	8B	4B	write allocate

a) For each cache configuration, fill in the fields in Table 5. (Entry E gives an example for a 4B, direct-mapped, 1B block size, no sub-block, no write-allocate cache.) **[8 pt]**

Table 5: Fields in the caches

	cache tag	cache index	# of valid bit(s) per block	total # blocks in cache	# sub-blocks per block
A	VA[15:5]	VA[4:3]	2	4	2
B	VA[15:4]	VA[3]	1	4	none
C	VA[15:5]	VA[4:3]	1	4	none
D	VA[15:4]	VA[3]	2	4	2
E	VA[15:2]	VA[1:0]	1	4	none

Left two columns: -1/2 pt. for every wrong answer
Right three columns: -1/2 pt. for every wrong answer up to -1 per row

b) Here is a sequence of ten one-byte memory references (in hex) to the caches:
R 0x0, W 0x4, R 0x6, R 0x20, R 0x25, W 0x27, W 0x1, R 0x23, R 0x3, R 0x4.
NOTE: R - read access, W - write access

Fill in hit or miss for each memory references for the four cache configurations. Compute the final cache miss rates. **NOTE:** each cache starts out empty. **[22 pt]**

Results for the above example, cache E, are shown in the last column.

Table 6: Results of the cache references

references	cache configurations				
	A	B	C	D	E
0x00 R	miss	miss	miss	miss	miss
0x04 W	miss	hit	hit	miss	miss
0x06 R	miss	hit	hit	hit	miss
0x20 R	miss	miss	miss	miss	miss
0x25 R	miss	hit	hit	miss	miss
0x27 W	hit	hit	hit	hit	miss
0x01 W	miss	hit	miss	hit	miss
0x23 R	hit	hit	miss	hit	miss
0x03 R	miss	hit	miss	hit	miss
0x04 R	miss	hit	hit	hit	miss
miss rate (%)	80%	20%	50%	40%	100%

-1/2 pt. for every wrong answer

Question 4: Pipelining

The designers are concerned about stalls in the pipeline. (Un)fortunately, the chief architect is out teaching. You are asked to examine the pipeline and recommend necessary bypasses to make the pipeline functional and to minimize pipeline stalls.

Assume a five-stage pipeline (IF, ID, EX, MEM, WB), with pipeline registers between adjacent stages. Pipeline registers are labeled T1 (IF/ID), T2 (ID/EX), T3 (EX/MEM), T4 (MEM/WB).

Fill in your bypass recommendations in Table 7. The table has a <pipeline register, opcode> pair in each column heading, and a <pipeline stage, opcode> pair for each row heading. The columns represent the source of bypasses. For example, column <T3, SUBU> is the output of EX stage latched in pipeline register T3 after executing SUBU. The rows represent the sink of the bypasses. For example, row <EX, LW> is the input to the ALU in EX stage for LW. **[20 pt]**

You may only forward to the components listed below:

IF	**ID**	**EX**	**MEM**	**WB**
Instr. Memory	**Comparator**	**ALU**	**Data Memory**	
	Sign Extend			
	PC			

Only add bypasses when it is needed for correctness or to prevent stalls. When no forwarding is needed, say so. You will lose points for adding unnecessary bypasses or leaving entries blank!

A few entries are labeled for you as examples. -1 for every wrong box

Table 7: Pipeline bypass

	<pipeline register, opcode>				
<stage, opcode>	T2 (ID/EX) BEQ	T3 (EX/MEM) SUBU	T3 (EX/MEM) SW	T4 (MEM/WB) LW	T4 (MEM/WB) JAL
IF, BEQ	No forwarding	No forwarding	No forwarding	No forwarding	No forwarding
ID, JR	No forwarding	T3 -> PC	No forwarding	T4 -> PC	T4 -> PC
ID, BNEZ	No forwarding	T3 -> Comp.	No forwarding	T4 -> Comp.	T4 -> Comp.
EX, JR	Not applicable	No forwarding	No forwarding	No forwarding	No forwarding
EX, LW	Not applicable	T3 -> ALU	No forwarding	T4 -> ALU	T4 -> ALU