# Department of Electrical and Computer Engineering The University of Texas at Austin

EE 460N Spring 2011 Y. N. Patt, Instructor Faruk Guvenilir, Milad Hashemi, Yuhao Zhu, TAs Exam 2 April 20, 2011

Name: $\int_{0}^{\infty} l u + i u n$
Problem 1 (30 points):
Problem 2 (20 points):
Problem 3 (25 points):
Problem 4 (25 points):
Total (100 points):
Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.
Note: Please be sure your name is recorded on each sheet of the exam.
Please sign the following. I have not given nor received any unauthorized help on this exam.
Signature: Solution

GOOD LUCK!

· · · · · · · · · · · · · · · · · · ·	
Name: Solution	
Problem 1 (30 points)	
Part a (6 points): A problem takes 2 hours to solve on the computer. It consists of two parts, which mu sequence. The first part is 90% parallelizable, the second part is 60% parallelizable. Show your work bel	that each pa st be done in takes ow. I have an
With infinite computing resources this problem would take how long?	sequentialmad
This represents a maximum speed-up of what? $(19) \cdot 60 \text{min} = 6 \text{ min}$ $(16) \cdot 60 \text{min} = 24 \text{min}$	
$\frac{4}{30} = 4$	
With 4 processors, the speed-up for this problem would be what?	
60×2 12 C	)
With 4 processors, the speed-up for this problem would be what? $ \frac{60 \times 2}{4} + (17) \cdot 60 + \frac{6 \cdot 60}{4} + (16) \cdot 60 = \frac{12 \cdot 00}{13.5 + 12.5} $ Part b (6 points): We have discussed in class various kinds of manners of this independent of the second of the seco	$6+9+24 = \frac{120}{52.5}$
Part b (6 points): We have discussed in class various kinds of memory, one of which is a content-addressory. Two structures we have discussed in class actually use content addressable memory. For each case, structure, and show all parts of an entry. Circle the element in each entry that is used for content addressing	Sable mem-
Name of structure: TLB	•
An entry in this structure: Page Number PTE	
Name of structure: Tag Store	

Part c (6 points): Classical VLIW machines do not make effective use of their instruction caches. Why? Explain.

NOPs that ourst be inserted into the assembly code waste space.

An entry in this structure:

Name: Solution

#### Problem 1 continued

Part d (6 points): We detect an interrupt or an exception; when does it get handled? We noted in class that interrupts are normally handled

When Convenient

whereas exceptions are normally handled

When Detected

However, as is often the case in computer architecture, "normally" is not the same as "always".

An example of an interrupt which is not handled at the time you stated above is

Machine Check

An example of an exception which is not handled at the time you stated above is

O verflow

Part e (6 points): Two device controllers A and B are on BR level #3. Device controller A is closer to the PAU. Suppose device controller A does not want the bus, but device controller B does. This is the highest priority bus request so the PAU grants the bus with BG #3. Before device controller A sees the BG signal, its device wants the bus. What does device controller A do when the BG signal arrives?

Takes the BG signal, does not poss it to BB, asserts SACK, and becomes the next bus master.

Joki This Problem
Uses the Bus
Described
In class.

Is this a race condition (yes/no) Explain.

No. A asserted BR before it received BG, so B does not get the BUS this cycle. There is no improper operation that results because of this.

Name: Solution

## Problem 2 (20 points)

Suppose IEEE decided to extend the Floating Point standard to include a 12 bit representation, keeping all the characteristics of the IEEE Standard intact. They assign bits to exponent and fraction such that the smallest representable positive number (i.e., smallest subnormal) is 1/1024 and the smallest normalized positive number is 1/16.

Part a: How many bits for exponent:

How many bits for fraction:

 $\frac{1}{16} \Rightarrow 1 \times 2^{-4} , 1-n = -1$   $\frac{1}{1024} \Rightarrow 0.000001 \times 2^{-4} , n = 5$ 

What is the value of n in "excess-n" code for exponents:

Part b: How do you represent "minus infinity":

1 11111 000000

Part c: How do you represent the largest normalized number:

163 × 225

163 × 230-5

Part d: Consider the value 139/128 times 2<sup>8</sup>. If we change how many bits we allocate to exponent and fraction, and we insist that the exponent 0 is representated by 011..1, can we represent this value exactly. If yes, identify the number of fraction bits, the number of exponent bits, and show the representation of this value:

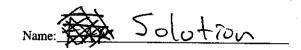
If no, explain why not, and represent the number according to the scheme of parts a,b,c above. If rounding is required, round to unbiased nearest.

139/128 = 1.0001011 -> 7 fraction bits, lenes 4 Exp. bits

Max Exponentis 1110 = 14; Excess is 7; 14-7=7

Thus, not possible.

0 0 1101 000110



Problem 3 (25 points)

\* Note: The Cache is Empty Beforehand

A byte-addressable cache of fixed total size and fixed line size is implemented both as a k-way set associative cache and also as a fully associative cache. Assume perfect LRU replacement and line allocate on a write miss policies. For this problem, please show your work below.

If the two implementations are presented with the same sequence of addresses for byte accesses, they generate hits/misses as shown below:

Address	R/W	k-way associative (Miss/Hit)	Fully associative (Miss/Hit)	7
011011100	W	Miss	Miss	<b>k</b>
011001100	R	Miss	Miss	
111000111	R	Miss	Miss	) hits because
011011000	R	Hit	Hit	hits because of this.
010101100	R	Miss	Miss	of this,
001111111	R	Miss	Miss	•
011001111	R	Miss	Hit	
111110010	R	Miss	Miss	
110100111	R	Miss	Miss	
111000011	R	Hit	Miss	

1- Byte on Block Bits

Part a: How many cache lines are there in the two cache implementations?

Part b: The set associative cache is k-way, where $k =$
1-Way (Direct Mapped)
2-Way
<b>3-Way</b>
4-Way
5-Way
6-Way
7-Way
8-Way
9-Way

(check the appropriate box).

Part c: For the k-way cache, how many tag bits: 5 and index bits:

Part d: Recall in the preamble, that this cache does an allocate on write miss. Complete the hit/miss behavior table if the cache did not allocate on a write miss.

Address	R/W	k-way associative (M/H)	Fully associative (M/H)
011011100	W	М	M
011001100	R	M	M
111000111	R	М	M
011011000	R	Μ	M
010101100	R	M	M
001111111	R	M	M
011001111	R	~	Н
111110010	R	<b>M</b>	~
110100111	R	М	
111000011	R	Н	M

Name: Solution

### Problem 4 (25 points)

Consider the following two level virtual memory system, similar to the VAX:

Virtual Address Space:

System Space Range:

512 Bytes

Physical Memory Size:

128 Bytes

User Space Range:

x000 to x0FF x100 to x1FF Page Size: PTE Size:

8 Bytes 1 Byte

The machine is byte addressable. Assume the system does not include a TLB. The PTE format is as follows:

Part a: How many bits are allocated for the PFN in the PTE?

$$\frac{PM \ Size}{Prye \ Size} = \frac{2^{2}}{2^{3}} = 2^{4} \triangle \text{Answer}$$

We wish to execute the instruction:

ADDM R2, R0, R1

where ADDM takes the contents of the memory location specified by R0, adds to it the value in R1, and stores the result in the memory location specified by R2. That is,

$$M[R2] < --M[R0] + R1$$

Before this instruction is executed, the following values are stored as shown:

R0: x0B9

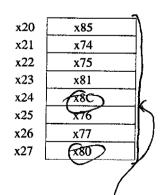
R1: x12

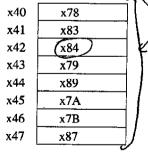
UBR (User Space Page Table Base Register): x140

must be part of systempage hable

## Memory:

x00	x70
x01	(x71)
x02	x8C
x03	x88
x04	x72
x05	x82
x06	x8F
x07	x73





x60	x8D	
x61	x8A	
x62	x86	
x63	x7C	
x64	x7D	
x65	x88	
x66	x7E	
x67	x7F	

mostbe part of user Page table.

#### **Problem 4 continued**

Part b: After the insruction was fetched and decoded, six subsequent memory accesses were required to process this instruction. There were no page faults, interrupts or other uhappy events.

Your job: Complete the table below:

Access #	VA	PA	Data	Description
1	NA	x42	×84	PTE for page Containing (SS)
2	x157	127	×80	PTE for page containing M[RO] (U
3	x089	10x	x71	Operand Data for ADDM
4	N/A	x42	x84	PTE for page containing 1 (SS)
5	x154	×24	x8C	PTE for page contany to pet som. (US)
6	× DAY	x64	×83-	Sum of MEROJ+RI

C sum: x71+x12=x83

Note: "N/A" is a potential answer for entries in the VA column. Data is the data that is Read from or Written to memory. Use the Description column to describe what is being read/written (e.g. "PTE for page xx," "Operand data for ADDM").

Part c: What is the value of the SBR (System Space Page Table Base Register)?

Answer: x 38

Part d: What is the maximum number of Page Faults ANY execution (excluding instruction fetch) of the ADDM instruction could generate?

x089 = 0 1011 1001 x17 +UBR= x17+ x140= x157 (157= 1 0101 0111 x0A+ SBR= x42, SBR= x42-x0A= x38 M[x92]= x84 = 1000 0000 000 x27 M[x27]= x80= [000 0000 + > x0]

x64 = 601100000 = Frame xC, lookat M[x24] \*24 on same page as x27, see M[x42]

System Page Table is always Resident in Physical Memory.

VA = R2 = x0 PN (00) = (from x64)

PN +UBR = PN +x140 = 61, PTE-PN, (60) (from x21 PTE-PN +SBR = PTE-PN+ x38 = x42, PTE-PN = xOA

b1 01010,100 = x154 PN+x140 = x154, PN=x154-x140 = x14