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

EE 460N, Spring 2012
Mattan Erez, Instructor
Exam 3, April 27, 2012

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam.

Signature : ______________________________

Printed Name : ______________________________

Question:	1	2	Total
Points:	50	60	110
Score:			

Note: Please be sure that your answers to all questions **(and all supporting work that is required)** are contained **in the space (boxes) provided**. Anything outside the boxes will be ignored in grading. Use the back of each page as scratch paper, if needed.

Note: "I DON'T KNOW" is a valid answer that automatically gives you 10% of the maximum possible grade for that problem. Be sure to either erase, or cross-out everything else and write I DON'T KNOW in large, capital letters. Anything else will result in us grading what you have. **A blank answer does not get you the 10%.**

Note: Please put your name at the top of each page of the exam and remember that anything outside a box will be ignored!

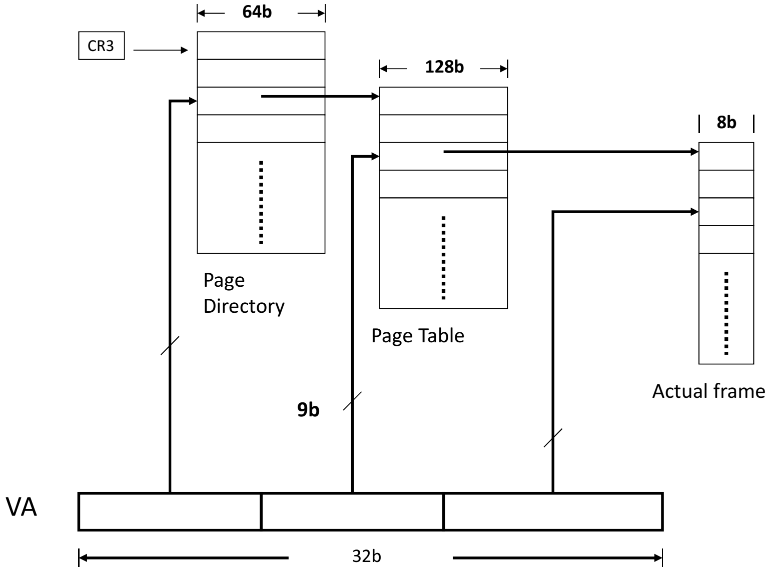
Note: For all questions, unless otherwise stated, find the most efficient (time, resources) solution.

GOOD LUCK!

Name: ____________________

1. Virtual memory and caching.

   (a) (10 points) Consider an address translation mechanism that is similar to the one of x86, but is not the same. The mechanism is shown in the figure below. What size of pages (frames) does this system support? Explain briefly.



Frames are either 8KB or 4MB in size.

Given a frame size, all pages at all levels are uniform in size. The size of a page of the "page table" is known: 2^9*16B = 8KB. Therefore, when all translation levels are used, the actual frame is 8KB, requiring 13 bits of address, and a page of the "page directory" uses 10 bits of address. When eliminating the middle translation, frames are now 2^22B = 4MB.

Name: ______________________________

Now consider a processor that uses VAX virtual memory (note the page size below) and has a cache and TLB:

- Byte-addressable with little-endian organization.
- The page size is 8KB.
- There are three TLBs, one for P0, one for P1, and one for the system segment.
- There is a two-level cache hierarchy. Caches are *virtually indexed and physically tagged*, have a line (block) size of 64 bytes (2^6B). There are separate first-level instruction and data caches, which are 32KiB (2^{15}B) each. There is a unified second-level cache that is 512KiB in size (2^{19}B).
- Main memory access time is $t_{mem} = 100ns$.
- TLB and cache access time is $t_{TLB} = t_{cache} = 10ns$.
- Cache, TLB, and memory accesses *are not* overlapped.
- `SBR = 0x00004000`, `P0BR = 0x80006000`, and `P1BR = 0x80003000`
- A PTE has the following format:

15 9	8	7 0
Don't care	V	PFN

The table below describes the content of some of the *physical* memory. Each cell in the table represents a single byte in hexadecimal, whose address is given by adding its row and column header values (header values in bold and also base 16).

(b) (20 points) Fill in the table below for translating the two addresses listed. Note that this is equivalent to what happens on a system, including multi-level translation, exceptions, ... If you need to explain your work, use the box on the next page.

The table entries have the following meaning:

VPN Virtual page number of the access.

PTE VA Virtual address of the PTE corresponding to the access.

PTE VPN Virtual page number of PTE(access).

PTE' PA The physical address in the system page table needed to locate PTE(access).

PTE PA The actual physical address of PTE(access).

PA The actual physical address of the access.

Data The value the access will return (1 byte).

Address	VPN	PTE VA	PTE VPN	PTE' PA	PTE PA	PA	Data
`0x00006004`	3	x80006006	3	x4006	x2006	x6004	1
`0x0000A00C`	5	x800600A	3	x4006	x200A	xA00C	N/A

Memory contents table:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	1	1	4	9	11	13	3	5	7	5	2	5	5	0	8	1D
1000	0	5	67	33	6	13	0	B	3	51	4	5	2	9	0	5
2000	4C	7	7	47	37	4	3	5	2	4	5	11	6	0	76	7
3000	0	5	7	8	6	5	E	B9	16	15	4	E	8	9	0	7
4000	4F	7	33	34	2	5	1	3B	42	5	1	F8	3	7	9	1
5000	27	1	5	51	0	5	7A	27	19	1	9	6	0	0	78	44
6000	0	5	99	19	1	4	0	7	58	85	23	17	51	6	1	1
7000	44	5	4	16	2	58	FF	8	45	9	8	37	0	FE	19	5E
8000	13	7	5	16	9A	3	21	19	19	0	96	0	4	0	0	5

Name: ____________________

Additional assumptions/explanation:

The PTE corresponding to the second address is invalid. This means there is a page fault and translation cannot be completed.

(c) (10 points) What is the minimum associativity the L1 data cache can have? Why?

Pages are 8KiB each and the cache is 32KiB in size, this means we need at least 4 ways (but can have more).

(d) (5 points) What size and associativity would you choose for P0TLB and P1TLB out of the following options? Explain very briefly. No points will be given to a poorly explained selection. As always, the idea is to balance tradeoffs and explain the choice.

	16-entry, fully associative
	32-entry, fully associative
	64-entry, fully associative
	64-entry, 16-way set associative
	128-entry, fully associative
	128-entry, 16-way set associative
	256-entry, fully associative
	256-entry, 16-way set associative
	1024-entry, fully associative
	1024-entry, 16-way set associative

	16-entry, direct mapped
	32-entry, 8-way set associative
	64-entry, 8-way set associative
	64-entry, direct mapped
	128-entry, 8-way set associative
	128-entry, direct mapped
	256-entry, 8-way set associative
	256-entry, direct mapped
	1024-entry, 8-way set associative
	1024-entry, direct mapped

All of physical memory is covered with 2^8 or 256 frames. So, 256 direct mapped is one reasonable answer. VPN is virtual though, so could have conflicts, so some associativity could be good. Smaller TLBs also OK, and no more than 8 or 16 associativity.

(e) (5 points) What size and associativity would you choose for STLB out of the following options? Explain very briefly. No points will be given to a poorly explained selection. As always, the idea is to balance tradeoffs and explain the choice.

	16-entry, fully associative		16-entry, direct mapped
	32-entry, fully associative		32-entry, 8-way set associative
	64-entry, fully associative		64-entry, 8-way set associative
	64-entry, 16-way set associative		64-entry, direct mapped
	128-entry, fully associative		128-entry, 8-way set associative
	128-entry, 16-way set associative		128-entry, direct mapped
	256-entry, fully associative		256-entry, 8-way set associative
	256-entry, 16-way set associative		256-entry, direct mapped
	1024-entry, fully associative		1024-entry, 8-way set associative
	1024-entry, 16-way set associative		1024-entry, direct mapped

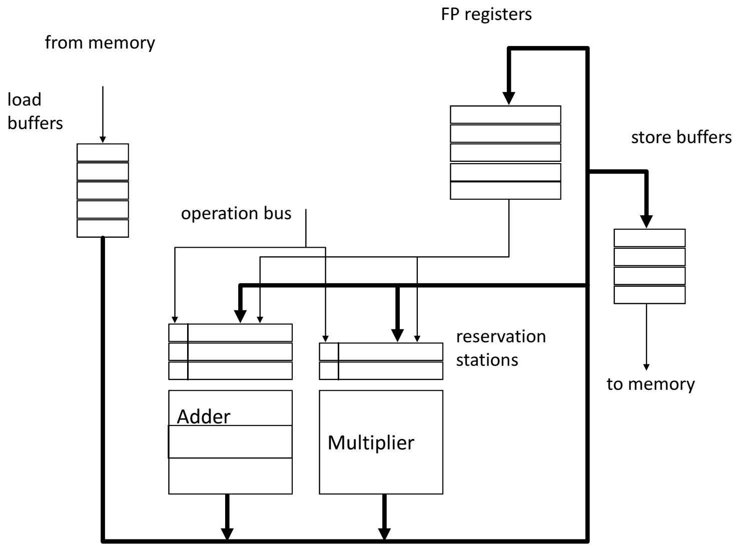
System should be using less memory than the applications so TLB should be smaller than for (f).

Name: ____________________

2. Pipelining and OOO execution.

(a) (10 points) How many rename tags does the following machine need? This is pretty much the original Tomasulo design of the floating-point pipeline of the IBM System 360 as discussed in class. Note that there is no ROB. Explain very briefly and state assumptions if any. Note that pipelined execution units are indicated by multiple "rectangles" – one per cycle. In general, everything in the figure has meaning, nothing was drawn arbitrarily.

(a) 14 or 15

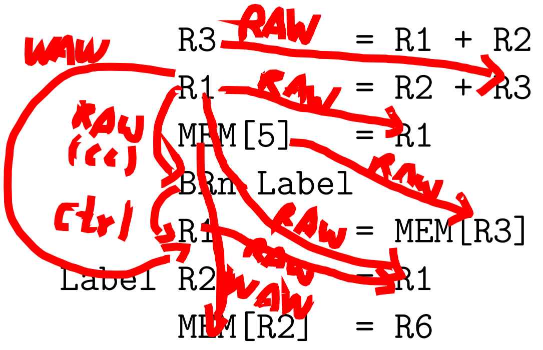


```
14 if assuming free-list managed in same cycle as WB.
15 if assuming free-list takes another cycle.

SB entries are for stores; stores do not require
rename tags because they don't write to a register.
```

Name: ______________________________

(b) (10 points) Identify *all potential hazards* in the following code. Mark dependencies with arrows and label the arrow with the type of dependence. Arrows should point to the instruction that executing OOO would cause a problem. This is independent of microarchitecture – identify all the potential issues.

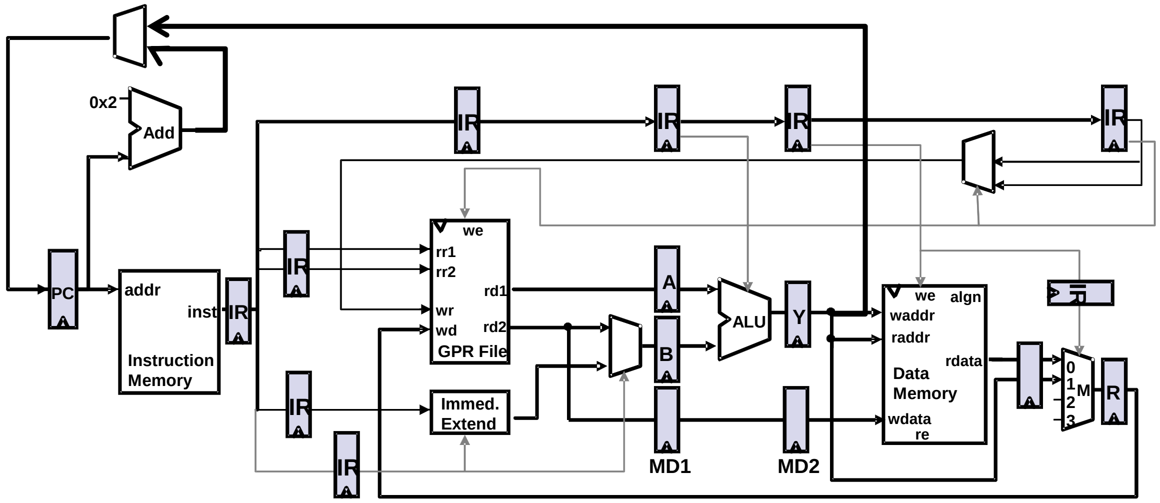


```
        R3      = R1 + R2
        R1      = R2 + R3
        MEM[5]  = R1
        BRn Label
        R1      = MEM[R3]
Label   R2      = R1
        MEM[R2] = R6
```

(c) (10 points) Consider the pipelined architecture below and fill in the stages in the slots below. A classic 5-stage pipeline will be F—D—E—M—W, but in the pipeline below some of the 5 stages may be split into multiple sub-stages for increased frequency and concurrency. Note that the table provides space for more stages than needed. Also note, that for simplicity, many of the control signals have been removed. The pipeline uses interlocks to stall on dependencies. State any assumptions in the box below the table.

F1	F2	D	E	M1	M2	W					

Could also interpret as decode taking two cycles or WB taking two cycles.



(d) (20 points) What would be the IPC (to within one decimal digit or written as a fraction) of the code below executing on the pipeline above? Explain your answer briefly and state any assumptions you make in the box. Be reasonable about the assumptions – if you're not sure whether an assumption you're making is reasonable or not, ask us. Assume that there are separate instruction and data caches and that the hit rate for both is 1. Note that the values of the registers shouldn't matter.

```
Label  R1     = R1 + 1
        MEM[R1] = R3
R3     = R4 * R5
R2     = MEM[R3]
R7     = R0 * R0
R8     = R1 * 2
        BRnzp Label
```

```
7/17 if assuming D can happen same cycle as WB,
7/19 if assuming need to repeat D stage to read WB value.
```

Bonus Question worth 10 points

(e) (10 points) Now, you are asked to optimize the execution of the code above. You are allowed to modify anything you wish, but you are not allowed to add any of the following: execution units and similar resources, registers, latches, and memories. You may also not remove any hardware or change the semantics of the program and the pipeline (e.g., OOO completion). What would you change, how, and why?

Name: ______________________________