Computer Architecture, EITF20
Final Exam

LUND UNIVERSITY
Electrical and Information Technology
Jan. 17, 2019

The exam consists of 5 problems with a total of 50 points.
Grading: 20 p $\leq$ grade 3 $<$ 30 p $\leq$ grade 4 $<$ 40 p $\leq$ grade 5

Instructions:

- Turn off and put away your mobile phone – **No mobile phones**
- You may use a pocket calculator and an English dictionary on this exam, but **no other aids**
- Please start answering each problem on a new sheet – **New problem $\Longrightarrow$ New sheet**
- Write your anonymous code on each sheet of paper that you hand in – **Code on each sheet**
- You must motivate your answers thoroughly.
  Show all your calculations.
  If there, in your opinion, is not enough information to solve a problem, you can make reasonable assumptions that you need in order to solve the problem.
  State your assumptions clearly!

GOOD LUCK :-)

Problem 1

The CPU performance equation is an important tool to analyze the CPU performance:

a) Please write down the CPU performance equation (CPU execution time), assuming single-core CPU with 2 levels of Cache (3)

b) Describe each of the following items/concepts concerning computer architecture. Where relevant also give an example. Describe how these techniques affect the CPU performance equation. (point out **ALL** the possible effects) (4)

1) Out-of-Order execution
2) Larger (L1) cache block
3) Reorder buffer
4) Critical word first

c) Which of the above 4 techniques are related to Meltdown attack and how it is (they are) related to Meltdown? (3)

Problem 2

Consider following part of an assembly–language program:

```
1: LW      R5, (R3)           ;R5 = Mem[R3]
2: LW      R6, 8(R3)          ;R6 = Mem[R3+8]
3: MUL     R5, R5, R6         ;R5 = R5 * R6
4: ADD     R3, R3, R2         ;R3 = R3 + R2
5: SW      R5, (R3)           ;Mem[R3] = R5
```

Basic architecture setup: A standard 5-stage pipeline architecture with stalling but no forwarding; One arithmetic functional unit (for both multiplication and addition); All instructions take one cycle in the EXE stage, **except** MUL and ADD, which take 5 cycles and 2 cycles in the EXE stage respectively (The EXE stage of MUL and ADD is not pipelined); Assume a perfect memory system (all memory accesses are cache hits).

a) Draw an instruction execution timing table (clock number vs instruction number with entries showing the pipeline stage, similar to the table below) for the above code snippet. (4)

	Clock cycle no.																
Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**...**
LW R5, (R3)																	
LW R6, 8(R3)																	
MUL R5, R5, R6																	
ADD R3, R3, R2																	
SW R5, (R3)																	

Now, improve the basic pipeline architecture with 2 arithmetic functional units and the Tomasulo technique with 2 reservation stations for each arithmetic functional unit and enough reservation stations and units for loads/stores. Assume the execution time for each individual instruction remains the same.

b) Describe and specify the Tomasulo implementation in the pipeline (Key concept and main changes in the hardware). (2)

c) Draw the instruction execution timing table (clock number vs instruction number with entries showing the pipeline stage, similar to the table below). (4)

	Clock cycle no.																
Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**...**
LW R5, (R3)																	
LW R6, 8(R3)																	
MUL R5, R5, R6																	
ADD R3, R3, R2																	
SW R5, (R3)																	

Problem 3

We are interested in comparing two proposed enhancements to a baseline processor to decide which design offers highest performance. All processors assume an average CPI of 1.5 together with cache. Assume instruction fetch has 100% cache hit rate (i.e., instruction cache is perfect). Each access between cache and main memory to read or write a cache block requires a setup time of 30 clock cycles, with each transfer of 32-bit data adding another 2 clock cycles penalty. The cache size for all 3 processor designs is 32,768 bytes, i.e. 32 kB.

Instruction mix	
instruction type	**% of all instructions**
load	24.55
store	10.45
uncond branch	3.50
cond branch	11.50
int computation	39.15
fp computation	10.85

1. The baseline processor runs at 3.0 GHz clock frequency, has a cache with 128 sets, associativity 1, uses Random replacement policy, write-back with an average of 20% dirty blocks, and no write-back buffer.
2. Based on the baseline processor, the first modification suggests to use the FIFO replacement policy and to add a write-back buffer (same percentage of dirty blocks as before) which can hold 80% of the write-back blocks (assume the content in the write-back buffer will not cause any CPU stalls). Moreover, the execution time for the integer computation has been decreased by 3 times (on average).
3. Based on the baseline processor, the second modified design suggests a cache with 32 sets, associativity 4 and 30% of dirty blocks for write-back, with a write buffer which can hold 20% of the write-back blocks. Moreover, due to improved implementation, all floating-point computations are executed 50% faster (on average).

Following cache miss-ratio (Random replacement) are expected depending on configuration:

Cache miss-ratio			
No. of	**Associativity**		
sets	**1**	**2**	**4**
16	0.79	0.55	0.28
32	0.60	0.31	0.14
64	0.36	0.15	0.08
128	0.22	0.09	0.06

Clock cycle time dependence on associativity is:

	Associativity		
	1	**2**	**4**
relative clock cycle time	1	1.06	1.25

- Assume that the processor cannot continue until the complete block is fetched into the cache on a cache miss. Which of the 3 designs will give best performance? (**show all calculation!**) **(10)**

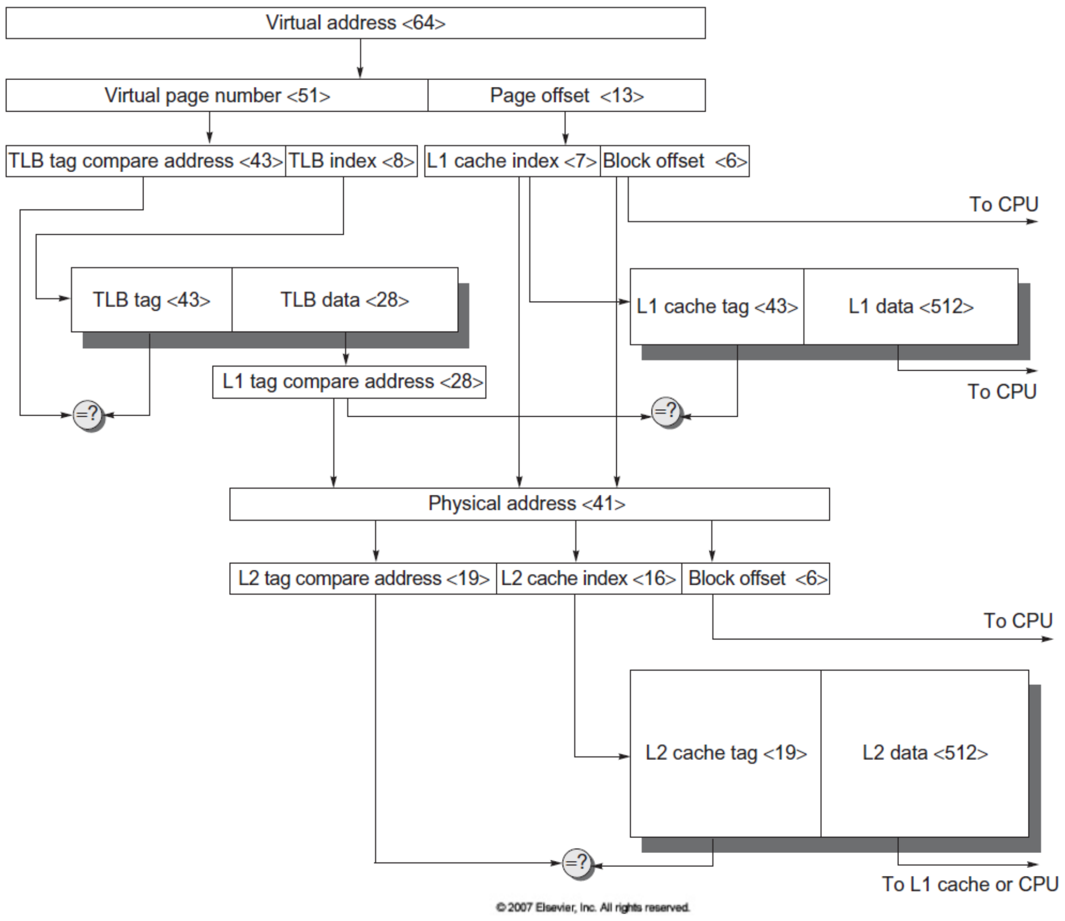
Problem 4

The following figure shows a hypothetical memory hierarchy going from virtual address to L2 cache access. The virtual address is 64-bit and the physical address is 41-bit. The corresponding address bit partitioning is illustrated in the figure (numbers in the ⟨⟩ are the number of bits).

a) Calculate Page size, L1 cache size, TLB set-associativity (TLB has 256 entries), and L2 set-associativity (L2 cache size = 8MByte). (4)

b) Briefly describe the operation steps required from virtual address to L2 cache access (assume access miss for L1 cache and TLB). (4)

c) Point out potential mistake(s) in the address partitioning in the figure and explain why it is (they are) wrong. (2)



Problem 5

Consider a multiprocessor system with one shared main memory and three processors, each with its own local cache.

a) The MESI protocol (**Fig. 1**) is used in a multiprocessor system to tackle the inconsistency problem. Consider the sequence of instructions listed in **Table 1**, which operate on the data located in a certain main memory address, i.e. "X". Fill in the table by denoting the MESI state in all caches, and the contents of "X" in caches and main memory after execution of each instruction. Note that the initial MESI state of all caches is "Invalid" and the initial content of "X" in the main memory is equal to **"5"**. (4)

b) Assuming the same hardware setup, a clean initial state (all caches in "Invalid" state), and the initial content of "X" in the main memory equal to **"5"**, can you come up with a series of instructions in the different processors that results in "X" being in a "Shared" state in all processors, with all caches and main memory having "X" with a value of **"7"**. Continuing from this state, add more instructions to have "X" only in P3 in a "Modified" state with its contents being **"9"**. (6)

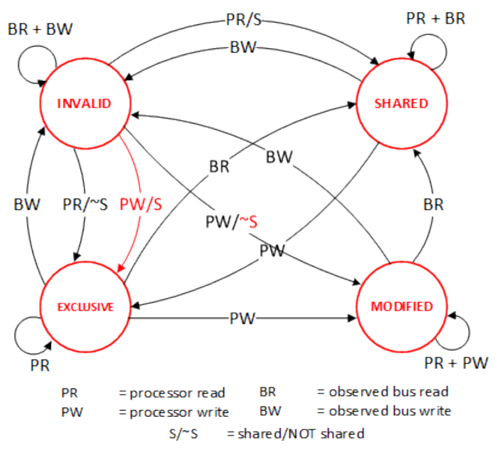


Figure 1: State diagram of MESI protocol.

	Processor1 (P1)		**Processor2 (P2)**		**Processor3 (P3)**		**Main Mem.**
Instruction	MESI Status	content of "X"	MESI Status	content of "X"	MESI Status	content of "X"	content of "X"
P2 Writes 1 to "X"							
P3 Reads "X"							
P1 Writes "7" to "X"							
P1 Reads "X"							
P1 Writes "8" to "X"							
P2 Reads "X"							
P3 Reads "X"							

Table 1: Running a sequence of instructions in the multiprocessor system (Problem 5).