國立清華大學 106 學年度碩士班考試入學試題

系所班組別：資訊工程學系 (0521)

考試科目（代碼）：計算機系統 (2102)

 *請在【答案卷、卡】作答

1. (6%) What is monolithic OS? What is microkernel OS? What is the kernel type of Apple's iOS and Google's Android? Please also give some examples to illustrate the services and/or functions found in a typical monolithic OS that may be external subsystems to a microkernel OS.

2. (4%) Please explain the concept of transaction atomicity.

3. (15%) A commonly used process model shows processes operating in the following five states: *new*, *ready*, *waiting* (or *blocked*), *running*, and *exit*, and the state transition events include admitted, interrupt (or timeout), completion, I/O (or event wait), dispatch and exit. But when considering the need of swapping, one *suspend* state and some state transition events will be newly added to the process model.

   (a) (4%) Please draw a six-state process model and indicate the types of events that lead to each state transition.
   (b) (7%) Please list all of the *impossible* transitions and explain why.
   (c) (4%) For the six-state process model, please draw a queuing diagram. Note that the queue obeys a First-In, First-Out (FIFO) rule.

4. (5%) Consider a system consisting of m resources of the same type that are shared by n processes, each of which needs at most k resources. What is the minimum numbers of instances of resources to guarantee that the system is deadlock free? Explain your answer.

5. (8%) A student in a compiler design course proposes to the professor a project of writing a compiler that will produce a list of page references that can be used to implement the optimal page replacement algorithm. Is this possible? Why or why not? Is there anything that could be done to improve paging efficiency at run time?

6. (6%) Is it possible for a process to have two working sets, one representing data and another representing code? Explain your answer.

7. (6%) Why might a system use interrupt-driven I/O to manage a single serial port, but polling I/O to manage a high performance network interface card?

8. (2%) What is the decimal value of "1111 0011", a one-byte 2's complement binary number?

9. (2%) For a color display using 8 bits for each of the primary colors (red, green, blue) per pixel and with a resolution of 1280x800 pixels, what should be the size (in bytes) of the frame buffer to store a frame?

10. (4%) Determine the values of the labels ELSE and DONE of the following segment of instructions. Assume that the first instruction is loaded into memory location $F0008000_{hex}$.

	slt	$t2, $t0, $t0
	bne	$t2, $zero, ELSE
	j	DONE
ELSE:	addi	$t2, $t2, 2
DONE:	…	…

11. (8%) Translate the following C code into MIPS assembly code:

    f = A[B[2]] - 4;

    Assume the following register assignment: f in $s1, the base address of A is at $s2 and that of B is at $s3.

12. (8%) You are using a tool that transforms machine code that is written for the MIPS ISA to code in a VLIW ISA. The VLIW ISA is identical to MIPS except that multiple instructions can be grouped together into one VLIW instruction. Up to *N* MIPS instructions can be grouped together (*N* is the machine width, which depends on the particular machine). The transformation tool can reorder MIPS instructions to fill VLIW instructions, as long as loads and stores are not reordered relative to each other (however, independent loads and stores can be placed in the same VLIW instruction). You give the tool the following MIPS program (we have numbered the instructions for reference below):

國立清華大學 106 學年度碩士班考試入學試題

系所班組別：資訊工程學系 (0521)

考試科目（代碼）：計算機系統 (2102)


```
(01) lw   $t0 ← 0($a0)
(02) lw   $t2 ← 8($a0)
(03) lw   $t1 ← 4($a0)
(04) add  $t6 ← $t0, $t1
(05) lw   $t3 ← 12($a0)
(06) sub  $t7 ← $t1, $t2
(07) lw   $t4 ← 16($a0)
(08) lw   $t5 ← 20($a0)
(09) srlv $s2 ← $t6, $t7
(10) sub  $s1 ← $t4, $t5
(11) add  $s0 ← $t3, $t4
(12) sllv $s4 ← $t7, $s1
(13) srlv $s3 ← $t6, $s0
(14) sllv $s5 ← $s0, $s1
(15) add  $s6 ← $s3, $s4
(16) add  $s7 ← $s4, $s6
(17) srlv $t0 ← $s6, $s7
(18) srlv $t1 ← $t0, $s7
```

(a) (2%) Draw the dataflow graph of the program. Represent instructions as numbered nodes (01 – 18), and flow dependences as directed edges (arrows).

(b) (1%) When you run the tool with its settings targeted for a particular VLIW machine, you find that the resulting VLIW code has 9 VLIW instructions. What minimum value of *N* must the target VLIW machine have?

(d) (2%) Based on the code above and the minimum value of N in (b), write down the MIPS instruction numbers corresponding to each VLIW instruction in the table below. If there is more than one MIPS instruction that could be placed into a VLIW instruction, please choose the instruction that comes earliest in the original MIPS program.

	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.
VLIW Instruction 1:										
VLIW Instruction 2:										
VLIW Instruction 3:										
VLIW Instruction 4:										
VLIW Instruction 5:										
VLIW Instruction 6:										
VLIW Instruction 7:										
VLIW Instruction 8:										
VLIW Instruction 9:										

(e) (1%) You find that the code is still not fast enough when it runs on the VLIW machine, so you contact the VLIW machine vendor to buy a machine with a larger machine width *N*. What minimum value of *N* would yield the maximum possible performance (i.e., the fewest VLIW instructions), assuming that all MIPS instructions (and thus VLIW instructions) complete with the same fixed latency and assuming no cache misses?

(f) (2%) Write the MIPS instruction numbers corresponding to each VLIW instruction, for this optimal value of *N*. Again, as in part (c) above, pack instructions such that when more than one instruction can be placed in a given VLIW instruction, the instruction that comes first in the original MIPS code is chosen.

	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.
VLIW Instruction 1:										
VLIW Instruction 2:										
VLIW Instruction 3:										
VLIW Instruction 4:										
VLIW Instruction 5:										
VLIW Instruction 6:										
VLIW Instruction 7:										
VLIW Instruction 8:										
VLIW Instruction 9:										

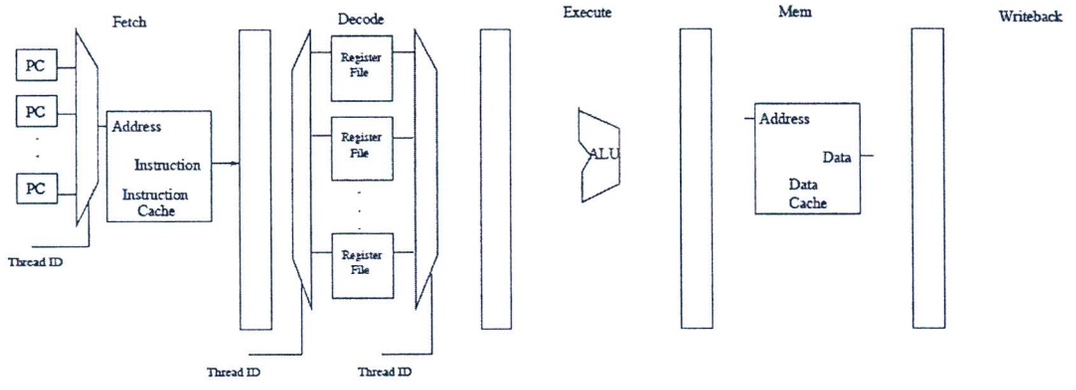
13. (9%) Fine-Grained Multithreading (FGMT): Consider a design "Machine I" with five pipeline stages: fetch, decode, execute, memory, and write back. Each stage takes 1 cycle. The instruction and data caches have 100% hit rates (i.e., there is never a stall for a cache miss). Branch directions and targets are resolved in the execute stage. The pipeline stalls when a branch is fetched, until the branch is resolved. Dependency check logic is implemented in the decode stage to detect flow dependences. The pipeline does not have any forwarding paths, so it must stall on detection of a flow dependence.

In order to avoid these stalls, we will consider modifying Machine I to use fine-grained multithreading.

國立清華大學 106 學年度碩士班考試入學試題

系所班組別：資訊工程學系 (0521)

考試科目（代碼）：計算機系統 (2102)



(a) (2%) In the five-stage pipeline of Machine I shown above, the machine's designer first focuses on the branch stalls, and decides to use fine-grained multithreading to keep the pipeline busy no matter how many branch stalls occur. What is the minimum number of threads required to achieve this? Why?

(b) (2%) The machine's designer now decides to eliminate dependency-check logic and remove the need for flow-dependence stalls (while still avoiding branch stalls). How many threads are needed to ensure that no flow dependence ever occurs in the pipeline?

A rival designer is impressed by the throughput improvements and the reduction in complexity that FGMT brought to Machine I. This designer decides to implement FGMT on another machine, Machine II. Machine II is a pipelined machine with the following stages.

Fetch	1 stage
Decode	1 stage
Execute	8 stages (branch direction/target are resolved in the first execute stage)
Memory	2 stages
Writeback	1 stage

Assume everything else in Machine II is the same as in Machine I.

(c) (2%) Is the number of threads required to eliminate branch-related stalls in Machine II the same as in Machine I? If YES, why? If NO, how many threads are required?

(d) (3%) Now consider flow-dependence stalls. Does Machine II require the same minimum number of threads as Machine I to avoid the need for flow-dependence stalls? If YES, why? If NO, how many threads are required?

14. (8%) Assume you developed the next greatest memory technology, MagicRAM. A MagicRAM cell is non-volatile. The access latency of a MagicRAM cell is 2 times that of an SRAM cell but the same as that of a DRAM cell. The read/write energy of MagicRAM is similar to the read/write energy of DRAM. The cost of MagicRAM is similar to that of DRAM. MagicRAM has higher density than DRAM. MagicRAM has one shortcoming, however: a MagicRAM cell stops functioning after 2000 writes are performed to the cell.

(a) (1%) Is there an advantage of MagicRAM over DRAM? Why?

(b) (1%) Is there an advantage of MagicRAM over SRAM?

(c) (3%) Assume you have a system that has a 32KB L1 cache made of SRAM, a 8MB L2 cache made of SRAM, and 2GB main memory made of DRAM, as shown in Fig. 1.

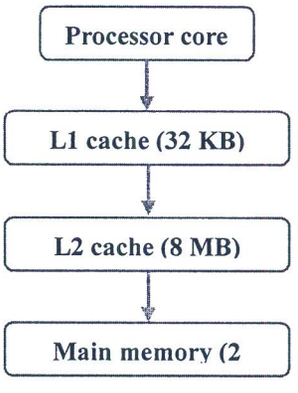


Fig. 1. Memory hierarchy of the system

Assume you have complete design freedom and add structures to overcome the shortcoming of MagicRAM. You will be able to propose a way to reduce/overcome the shortcoming of MagicRAM (note that you can design the hierarchy in any way you like, but cannot change MagicRAM itself).

Does it make sense to add MagicRAM somewhere in this memory hierarchy, given that you can potentially reduce its shortcoming? If so, where would you place MagicRAM? If not, why not? Explain below clearly and methodically. **Depict in a figure clearly and describe why you made this choice.**

(d) (3%) Propose a way to reduce/overcome the shortcoming of MagicRAM by modifying the given memory hierarchy. **Be clear in your explanations and illustrate with drawings** to aid understanding.

15. (9%) Consider the following three processors (X, Y, and Z) that are all of varying areas. **Assume that the single-thread performance of a core increases with the square root of its area.**

Processor X	**Processor Y**	**Processor Z**
Core area = A	Core area = 4A	Core area = 16A

(a) (2%) You are given a workload where **S** fraction of its work is serial and **1-S** fraction of its work is in infinitely parallelizable. If executed on a die composed of 16 Processor X's, what value of **S** would give a speedup of 4 over the performance of the workload on just one Processor X?

(b) (2%) Given a homogeneous die of area 16A, which of the three processors would you use on your die to achieve maximal speedup? What is that speedup over just a single Processor X? Assume the same workload as in part (a).

(c) (2%) Now you are given a heterogeneous processor of area 16A to run the above workload. The die consists of 1 Processor Y and 12 Processor X's. When running the workload, all sequential parts of the program will be run on the larger core while all parallel parts of the program run exclusively on the smaller cores. What is the overall speedup achieved over a single Processor X?

(d) (2%) One of the programmers decides to optimize the given workload so that it has 10% of its work in serial sections and 90% of its work in parallel sections. Which configuration would you use to run the workload if given the choices between the processors from part (a), part (b), and part (c)? Please write down the speedups for the three configurations.

(e) (1%) Typically, for a realistic workload, the parallel fraction is not infinitely parallelizable. What are the three fundamental reasons?