UNIVERSITY OF COLOMBO, SRI LANKA

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY (EXTERNAL)

Academic Year 2017 – 3rd Year Examination – Semester 5

IT5305: Computer Systems II

Structured Question Paper

June 9th, 2017

(TWO HOURS)

To be completed by the candidate

BIT Examination Index No: ....

Important Instructions:

- The duration of the paper is **2 (two) hours**.
- The medium of instruction and questions is English.
- This paper has **5 questions** and **11 pages**.
- **Answer all questions.** Questions **do not** carry equal marks.
- **Write your answers** in English using the space provided **in this question paper**.
- Do not tear off any part of this answer book.
- Under no circumstances may this book, used or unused, be removed from the Examination Hall by a candidate.
- Note that questions appear on both sides of the paper.
  If a page is not printed, please inform the supervisor immediately.

Questions Answered

Indicate by a cross (×), (e.g. ☒) the numbers of the questions answered.

To be completed by the candidate by marking a cross (×).	1	2	3	4	5	
Marks allocated for each question	19	18	25	24	14	
To be completed by the examiners:						

1) (a) A computer system has a CPU with a word length of 64 bits and a 48 bit virtual address. The virtual address is translated into a 34 bit physical address. Virtual memory (VM) is paged with 4Mbyte pages and the cache is 1Mbytes. The disk is 100Gbytes.

(i) What is the main memory (PM) size in Mbytes? **[2 marks]**

ANSWER IN THIS BOX

Main memory = 2^{34} bytes, which is approximately 2^{14} Mbytes given that 1 Mbyte = 2^{20} bytes

(ii) What fraction of VM pages can be there in the PM at any given time? **[3 marks]**

ANSWER IN THIS BOX

Since page sizes are same, fraction = PM size/ VM size

So, $2^{34}/2^{48} = 2^{-14}$

(iii) How many CPU words can be in a page? **[2 marks]**

ANSWER IN THIS BOX

Page size is 2^{22} bytes and a word is 2^6 bytes. Therefore, 2^{16} words in a page.

(b) A certain computing cluster has 8 homogeneous (identical) nodes, each with 8 processors. Each processor has a 1GHz clock and a CPI (cycles per instruction) of 0.3. Assume that each computing operation (integer, floating point etc) requires a single instruction. Amdhal's Law: speed up = n/{f + n(1-f)} where f – parallelizable fraction of computation, n – number of processors.

(i) What is the 'operations per second' rate of a node containing 8 processors? **[3 marks]**

ANSWER IN THIS BOX

Each processor IPS = cycles/sec divided by cycles/instruction

Each processor IPS = 10^9x8/0.3

For all 8 CPU per node = 64X10^9/0.3 operations per second

(ii) What is the 'operations per second' rate of the whole cluster containing 8 nodes? **[2 marks]**

ANSWER IN THIS BOX

For cluster Ops/sec = 8x64x10^9/0.3

(iii) If a task that is fully parallelizable is run on the cluster, what is the speedup achievable over a single processor? **[3 marks]**

ANSWER IN THIS BOX

If fully parallisable, then f=1 in Amdahl's law. Then speed up = 8x8 = 64.

(iv) Using Amdhal's Law, find out the inherent parallelism that a task should have to obtain a speedup of 32. **[4 marks]**

ANSWER IN THIS BOX

Rewriting Amdahl, we get f = (ns – n)/(ns –s).

So, f = (64x32 – 64)/ (64x32 – 32) = 62/63.

2) Consider the following high level code fragment. Here, the array of 4 byte integers a[0], a[1],..a[9] is stored starting at memory location 500 (that is a[0] at 500, a[1] at 504 and so on). The constant b is stored at memory location 700. The final result of the array a[ ] should be stored at the same location as the initial array. Your code should start at memory location 2000.

```
for i=0 to 9
  if (a[i] < b) then
      a[i] = 0;
```

(a) Assuming that the processor has only one internal register R, initialized to zero, write down the machine instruction sequence corresponding to a Register/Memory architecture. Your code should have not more than 8 instructions. The typical instruction set for a R/M architecture is as follows: **[9 marks]**

```
 LOAD M : [R is loaded the content of memory location or
pointer M]

 STORE M: [R is stored at memory location or pointer  M]

 ADD M: [adds R to memory content at M and saves in R]

 SUB M: [subtracts R from memory content at M and saves
 result in R and raises a flag if result is GT/LT/EQ/GE ]

 COMPARE M1, M2: [compares memory contents at locations(or
 pointer) M1 and M2 and raises a flag if M1 GT/LT/EQ/GE to
 M2]

 JUMP_flag M: [jumps to memory address(or pointer) M if flag
 is raised] (flags: GT - greater than 0; EQ – equal to zero;
 GE – greater than or equal to zero)

 JUMP M: [jumps to memory address(or pointer) M]

 SET M <target>: set memory location pointer M to "target"
 address

 INC (or DEC) M: increment (or decrement)memory location
 pointer M by 4
```

ANSWER IN THIS BOX

2000 SET M, 536

2001 COMPARE M, 700

2002 JGT 2004

2003 STORE M

2004 DEC M

2005 COMPARE M, 500 : there is a mistake here. There should have been an instruction to compare memory pointers?

2006 JNEZ 2001

This part can be given full marks if attempted.

(b) Assuming that the processor has a RISC architecture, write down the machine instruction sequence for the above high level code. Content of register R_0 is zero. Your code should have not more than 8 instructions. The typical instructions for a RISC architecture are as follows:

[9 marks]

LOAD R_x, offset[R_0]: [Rx is loaded from memory location (offset+R_0)]

ADD R_z,R_y,R_x : [Add contents of Rx to Ry and store it in Rz]

SUB R_z,R_y,R_x : [subtract contents of Rx from Ry and store it in Rz]

SUB R_z,R_z,#4: [subtract 4 from Rz and store it in Rz]

STORE offset[R_0],R_x : [Rx is stored at memory location (offset+R_0)]

JUMP_flag M: [jumps to memory address M if flag is raised] (flags: GT - greater than 0; EQ – equal to zero; GE – greater than or equal to zero)

JUMP M: [jumps to memory address M]

ANSWER IN THIS BOX

```
2000 ADD R1, R0, #36
2001 LD R2, 700[R0]
2002 LD R3, 500 [R1]
2003 SUB R4, R3, R2
2004 JUMP GT 2007
2005 SUB R1, R1, #4
2006 JUMP 2002
2007 ST 500[R1], R0
2008 JUMP 2002
```

3) In each of the sentences, underline or circle the more appropriate term to fill in the blanks, from the given options. **[25 marks]**

(i) In …(RISC/~~CISC~~) architectures, due to …… (constant/~~variable)~~ length instructions, instructions can be pre-fetched from memory without waiting for the current instruction to be decoded.

(ii) The availability of a large …(~~cache~~/number of registers) allows RISC architectures to reduce memory access bottleneck by keeping frequently used variables there.

(iii) …..(instruction pipelining/~~having complex instructions~~) allow(s) a processor to reach a cycles per instruction (CPI) of …(~~less than one~~/equal to one).

(iv) For a running process or a thread, if an instruction or a data fetch causes a cache miss but is located in …(main memory/disk memory), that process or the thread is ….(context switched/stalled) until the required instruction or data is fetched.

(v) A memory hierarchy consisting of cache, main memory and secondary memory work because some of the memories are …(~~fast and cheap~~/ fast and expensive) where as others are …(slow and cheap/~~slow and expensive~~), and also due to the code's property of …(locality of reference/~~Moore's law~~).

(vi) A computer's physical memory is always …(byte/~~word~~) organized whereas the logical memory fits in with ….(~~cache line size~~/register length).

(vii) A processor with a stack instruction architecture heavily utilizes the ….(~~cache~~/main memory).

(viii) According to ….(Flynn's~~/Moore's)~~ classification, the ….(~~MISD~~/SIMD) architecture is most efficient in processing arrays or vectors.

(ix) According to Amdhal's Law, a computing cluster with 16 processors will not give 16 times the speed up of a single processor because.... (~~of the memory bottleneck~~/the task has less inherent parallelism).

(x) In the standard 5 stage RISC instruction pipeline, the respective stages which are after ID and MEM stages are …(~~IF~~/EX) and…(WB/~~EX~~).

(xi) In a multitasking operating system, a process can be in one of three states: running, blocked or………….(~~killed~~/ready to run) state.

(xii) A thread is known as a………..(light weight/~~heavy weight)~~ process and multiple threads share same code and ……..(stack/~~data~~) but not ……………(~~stack~~/data).

(xiii) When a running process causes a page fault, it is moved to the ………….(blocked/~~killed~~) state and a waiting process is moved to the running state.

(xiv) To implement mutual exclusion for shared memory access by two or more threads, a spin-lock or a …………….(busy-wait/~~monitor)~~ can be used based on test-and-set machine instructions.

(xv) A hypervisor enables multiple guest operating systems to be run on the same piece of …………(processor hardware/~~network hardware~~).

4) (a) The C structure *task_struct* is defined in the *<linux/sched.h>* include file in the kernel source code directory. What is the significance of the *task_struct* structure? **[4 marks]**

ANSWER IN THIS BOX

It is the data structure that is used by the Linux operating system to store the details of each process.

(b) "Preemptive scheduling can result in race conditions when data is shared among several processes". Explain this observation. **[5 marks]**

ANSWER IN THIS BOX

Consider the case of two processes that share data. While one process is updating the data it may be preempted to allow another process to run. The second process then may try to read the data which is in an inconsistent state. That is a race condition.

(c) Assuming that the following process creation is successful, describe the outcome of the following C function. **[8 marks]**

```
int main(){

    int pid, i;

    for (i=0; i<2; i++) {
         pid = fork();
    }

    printf(“Hello! \n”);
}
```

ANSWER IN THIS BOX

At each iteration, a process will get created as a result of the *fork* function. Thus there will be four processes in total resulting in four *Hello!*'s being printed.

(d) Using the keywords *page table*, *physical address* and *page replacement algorithm* among others, describe the sequence of actions that take place when the CPU requires data from a *logical address* for a process. **[7 marks]**

ANSWER IN THIS BOX

1. The process *page table* is checked to see whether there is a valid mapping for the *logical address* to a *physical address.*

2. If yes, get the relevant data from the *physical address* in memory.

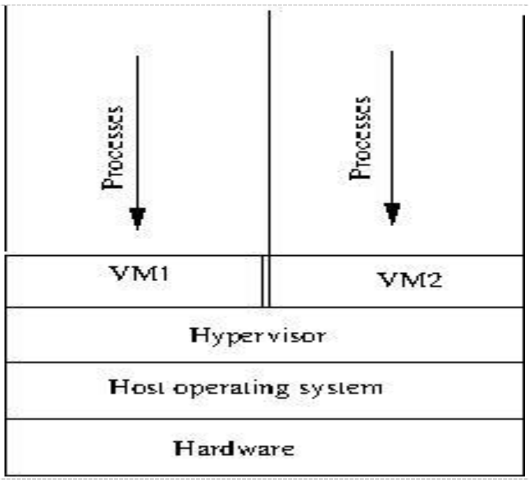
3. If not, and if there is a vacant frame in memory, bring the page to that frame and update the *page table.*

If no vacant frame in memory, use the *page replacement algorithm* to replace a page with the new page and update the *page table.*

5) **(a)** (i) Using a suitable illustration, describe the concept of *operating system virtualisation* clearly showing the position of the hardware, hypervisor and host operating system. **[5 marks]**

ANSWER IN THIS BOX

The *host operating system* runs on the *hardware* of a computer. The *hypervisor* runs on the *host operating system.* The *hypervisor* creates and runs virtual machines to enable multiple operating systems to run. A single physical machine can thus run multiple operating systems concurrently, each in its own virtual machine.



(ii) Name two popular hypervisors. **[3 marks]**

ANSWER IN THIS BOX

VMware ESX
Xen Server
Oracle Virtual Box
IBM LPAR
Oracle LDOM
Joyent Smart OS

(b) (i) Briefly explain why *cloud computing* is called an ‘on demand’ computing service paradigm. **[3 marks]**

ANSWER IN THIS BOX

In cloud computing, resources (such as storage capacity, computing power, and software applications) are provided to users on an *as-needed and when needed basis* resulting in it being called an *on-demand computing service paradigm.*

(ii) Name three types of cloud computing services giving one practical example of each. **[3 marks]**

ANSWER IN THIS BOX

1. Software as a Service (applications available via the internet)

Examples: Google Apps, Microsoft office365, Google docs, Gmail

2. Platform as a Service (a software stack ready for application use via the internet)

Examples: Force.com, Google App Engine

3. Infrastructure as a Service (networking, servers or storage available over the internet)

****