UNIVERSITY OF DUBLIN

TRINITY COLLEGE

Faculty of Engineering and Systems Sciences
Department of Computer Science & Statistics

Integrated Computer Science Programme
Integrated Engineering
Annual Examinations

Trinity Term 2014

CS3201 – Computer Architecture II

Tuesday, 6th May 2014 **Exam Hall** **14:00 – 16:00**

Dr Jeremy Jones

Instructions to Candidates:

Answer **THREE** questions.

Materials Permitted:

None.

Q1. Briefly describe the IA32 and x64 procedure calling conventions. Make sure that you describe the register set, draw a diagram of a typical procedure stack frame, list the volatile registers and in the case of the x64 calling convention explain the use of the *shadow space*.

[6 Marks]

What is the main advantage of the x64 procedure calling convention compared with the IA32 procedure calling convention?

[2 marks]

Convert the following code segment into (i) IA32 and (ii) x64 assembly language (assume INT is defined as a 32 bit integer for IA32 code and as a 64 bit integer for x64 code):

```
INT p2(INT a, INT b)
{
   return a + b;
}

INT p5(INT a, INT b, INT c, INT d, INT e)
{
   return a + b + c + d + e;
}

INT r0 = p3(1, 2, 3);
INT r1 = p5(1, 2, 3, 4, 5);
```

[12 marks]

Q2. How are virtual addresses converted to physical addresses by a two level page table? Assume a page table structure as per an IA32 based CPU. Draw appropriate diagrams to illustrate your answer.

[4 marks]

What is the advantage of using a two level page table compared with a single level page table? Comment on the amount of physical memory needed for the page tables of (i) a "small" process and (ii) a maximum sized process using a single level and a two level page table. Draw appropriate diagrams to illustrate your answer.

[4 marks]

A user process is created which has 0x7000 bytes of code, 0x1678 bytes of initialised data, 0x2888 bytes of uninitialised data and 648 bytes of stack data copied from its parent. Draw a diagram that shows the structure and size of the two level page table which is initially created for the process. What size is the initial memory foot print of the processes?

[5 marks]

(Question 2 continues on next page)....

.... (Question 2 continued from previous page)

If the following valid memory accesses are made by the process, draw a diagram which shows the changes that would be made to the initial page table structure and memory foot print as a result of handling any page faults. Provide a brief explanation for each change.

```
U     0x00001000
U     0x00001004
U     0x00001008
U     0x00002000
U     0x00007000
U     0x00009000
U     0xFFFFE800
U     0xFFFFD800
U     0x00002000
U     0x00802000
U     0x00002000
U     0x00009020
```

[7 marks]

Q3. What is a cache? How does a cache reduce the effective memory access time?

[2 marks]

With the aid of a diagram, explain how a cache organisation can be characterised by the three constants LKN show in detail how a data item is accessed in an LKN cache.

[6 marks]

Compute the number of hits and misses if the subsequent list of hexadecimal addresses is applied to

(1) a 128 byte direct mapped cache with 16 bytes per line and

(2) a 96 byte 3-way set associative a cache with 16 bytes per line.

0x0000 → 0x0004 → 0x000c → 0x0020 → 0x0028 → 0x0020 →
0x0208 → 0x0004 → 0x0304 → 0x0208 → 0x0020 → 0x020c →
0x0000 → 0x0020 → 0x0014→ 0x0080

Assume that (i) the lower 4 bits is used as the offset into the cache line and the next $\log_2(N)$ bits select the set (ii) all cache lines are initially empty/invalid and (iii) a LRU replacement policy.

[12 marks]

Q4. What is the cache coherency problem? Briefly explain the states and operation of the MESI cache coherency protocol.

[6 marks]

Consider a three CPU multiprocessor system where each CPU has its own cache. Each cache is direct mapped with two sets (i) even addresses map to set 0 and odd addresses map to set 1 and (ii) the caches are initially empty. Explain in detail (e.g. bus traffic and state transitions) what happens when each of the following "memory" accesses are made by the CPUs.

t	CPU 0	CPU 1	CPU 2
0	read a0		
1	read a0		
2	write a0		
3	write a0		
4		read a0	
5			read a0
6			write a0
7		write a0	
8	write a2		
9	write a0		
10	write a0		

[10 marks]

Assuming the multiprocessor described above, imagine that each CPU can execute a tight loop which continuously increments a shared variable *v*. Comment on the rate at which *v* is incremented if 0, 1, 2, or 3 CPUs are concurrently executing the loop. Assume a round robin bus arbiter.

[4 marks]