FIT9137 Workshop

Week 3

Topics:

- Introduction to memory design and management.

Covered Learning Outcomes:

- Describe basic concepts of computer memory management.

Instructions:

- One of the main targets of workshops is to anchor the learner into the session and create many opportunities to reinforce the learning in different ways – individually and in small groups. Sometimes we also teach key practical/theoretical concepts to you during these sessions.
- Form groups of 4-5 students to work through the exercises. If you meet a problem, try to solve it within your group by discussing it with your group members. If not resolved within the group, ask one of the support tutors to help you.
- You still have a question? Jump into one of many consultation hours run by our experienced tutors and seek help. Please visit the "Teaching Team and Unit Resources" tile in the FIT9137 Moodle site.

Activity A: Random Access Memory (RAM) Design using Memory Chips

In this task, we will analyze the structure of a 2K Byte (211X8) Memory Chip using some schematic diagrams. This memory chip is designed as 2K different memory locations of size 1 Byte, thus we can say 2K Byte = 2^{11} X 8 bit. This memory chip needs 11bits in the address line to select one byte of this memory. We also need a decoder to select one out of 2^{11} = 2048 Bytes that are available in this memory. Draw a schematic diagram of this memory. Design a 1-Mbit Random Access Memory (RAM) using 2K Byte (211X8) Memory Chip.

Activity B: Memory Management - Paging & Page Replacement Algorithms

During memory access, a **hard fault** occurs when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory. This new requested page needs to be brought into the memory slots, and it often requires replacing a page that is already in the memory from a previous request. In this activity, we will use two

page replacement algorithms that are used in operating systems: (i) First-In-First-Out (FIFO) (ii) Least-Recently-Used (LRU).

1. First-In-First-Out (FIFO) Page Replacement Algorithm

In this task assume you have 3 slots (or frames) in your memory. Consider a sequence of pages written on top of the below table. Find the number of hard faults if one uses FIFO algorithm to replace pages into memory. In the FIFO algorithm, we will replace a page (one out of the three in the memory) that came into the memory the earliest. Let's look at the table below showing page request sequences (1st row) and the empty memory slots or frames.

Page -> Request	3	2	1	0	3	2	4	3	2	1	0	4
Frame1->												
Frame2->												
Frame3->												

Then, we fill the first 3 columns to initiate the process for you. Initially all slots are empty. So, at the first 3 instances of page requests, when requests for pages 3, 2, 1 came they were allocated to the empty slots. This also gives us 3 Page Faults.

Page -> Request	3	2	1	0	3	2	4	3	2	1	0	4
Frame1->			1									
Frame2->		2	2									
Frame3->	3	3	3									

A. At the 4th instance, when a request for Page 0 comes, it is not available in memory, what happens? We have to remove one of the pages (which one? Use FIFO logic here) in the memory and store page 0 there. Is it a **Page Fault?**
B. And the requests for pages continue and you will experience more page faults. How many page faults in total?

2. Least-Recently-Used (LRU) Page Replacement Algorithm

In this task assume you have 3 slots (or frames) in your memory. Consider a sequence of pages written on top of the below table. Find the number of hard faults if one uses LRU algorithm to replace pages into memory. In the LRU algorithm, we will replace a page (one

out of the three in the memory) that was not recently used (or requested). Let's look at the table below showing page request sequences (1st row) and the empty memory slots or frames.

Page -> Request	3	2	1	0	3	2	4	3	2	1	0	4
Frame1->												
Frame2->												
Frame3->												

Then, we fill the first 3 columns to initiate the process for you. Initially all slots are empty. So, at the first 3 instances of page requests, when requests for pages 3, 2, 1 came they were allocated to the empty slots. This also gives us 3 Page Faults.

Page -> Request	3	2	1	0	3	2	4	3	2	1	0	4
Frame1->			1									
Frame2->		2	2									
Frame3->	3	3	3									

A. At the 4th instance, when a request for Page 0 comes, it is not available in memory, what happens? We have to remove one of the pages (which one? Use LRU logic here) in the memory and store page 0 there. Is it a **Page Fault?**
B. And the requests for pages continue and you will experience more page faults. How many page faults in total?

3. Increase the number of memory slots

In this task, let us assume we have 4 slots in your memory. Repeat tasks 1 and 2 and count the number of hard faults. Compare the results and note any counterintuitive outcome. The phenomenon is known as **Bélády's anomaly** which explains that not always increasing resources will bring better results. Do a little bit of research on **Bélády's anomaly.**