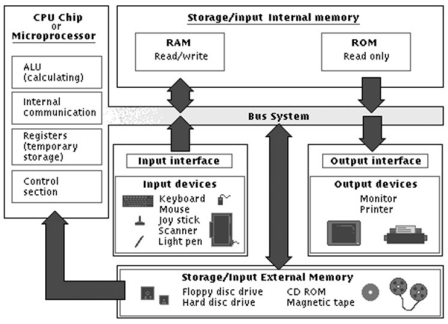
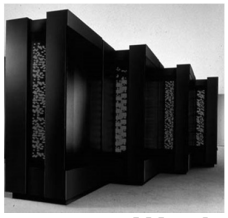
COMP25111
Operating Systems
Lectures 11



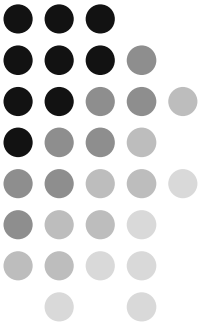
Dr Richard Neville
r.neville@manchester.ac.uk
Room: G12 Kilburn Building, Bottom floor



Week

7

NOTE: The **up-to-date** version of this lecture is kept on the associated web site – available [on-line] @ Blackboard select: COMP25111 Introduction to Computer Systems **www.manchester.ac.uk/portal**



This week's Short Exam Questions Q1

1. Question

   What is the difference between a '*partition*' and a '*program*.'

   ANSWER(S):

   a) A partition is: division of the storage area of a memory.

   b) A program is: supplies a computer with a set of pre-written instructions.

- NOTE: In the exam approximately 2 question are taken from the topics (and program examples) coved in each lecture

This week's Short Exam Questions Q3

1. Question

What is the difference between a '*partition*' and a '*program*.'

ANSWER(S):

a) A partition is: division of the storage area [memory] of a memory (Computers) .
A reserved part [block] of memory that is set aside for some purpose;

b) A program is: a program supplies a computer or other machine with a set of pre-written instructions (also programme) .
A collection of instructions that tell the computer what to do.

2. Question

What is a '*fixed partition*':

ANSWER(S):

Fixed partition divides memory into fixed size blocks..

3. Question

Differentiate between fragmentation and compaction:

a) Fragmentation: xxx. B) Compaction: xxx.

ANSWER(S):

a) Fragmentation: Storing programs (data) in non-contiguous areas in memory. As programs (data) are swapped, new programs (data) are stored in available free space - memory space, which may not be contiguous. Fragmented memory cause areas of memory that is not used; nominally these are small areas. A compaction program (algorithm) is used to move memory block so there are no spaces [fragments].

b) Compaction: memory compaction (block compaction) Any of several methods used to relocate information blocks in main memory in order to maximize the available free space in the memory.

- NOTE: In the exam approximately 2 question are taken from the topics (and program examples) coved in each lecture

Questions

Introduction to Questions:

The set of questions are based on lecture 11.

Answer Sheet will be given later in year and will contain the answers to these questions.

- Remember to find detailed and comprehensive answer you should [also] reference associated text books in the library.
- A reasonable starting place for associated book titles are:

1) This units 'module guide'; given to you in RN's first lecture – or on the web [Blackboard];
2) Those books mentioned in 'Background Reading;'
3) Those books [and web resources] mentioned in Learning Resources.

1. Question:

- Given the following address spaces and associated page sizes; calculate the number of pages that result in the *virtual address space*:

a) Page size 32KB and virtual address space 2GB;

b) Virtual address space 8GB and page size 256KB.

Answer:

1. Answer:

- Given the following address spaces and associated page sizes; calculate the number of pages that result in the *virtual address space*:

a) Page size 32KB and virtual address space 2GB;

If the virtual address space is 2 GB and the page size is 32 KB there are:

$$\text{Number of Pages} = \frac{\text{Address space}}{\text{Page size}}$$

$$= \frac{2\ \text{GB}}{32\ \text{KB}} = \frac{2^{31}}{2^{15}} = \frac{2{,}147{,}483{,}648}{32{,}768} = 64\text{k (32K) pages.}$$

[subtract powers 31-15=16; 2^{16}= 65,536 =~64k]

b) Virtual address space 8GB and page size 256KB.

If the virtual address space is 8 GB and the page size is 256 KB there are:

$$\text{Number of Pages} = \frac{\text{Address space}}{\text{Page size}}$$

$$= \frac{8\ \text{GB}}{256\ \text{KB}} = \frac{2^{33}}{2^{18}} = \frac{8{,}589{,}934{,}592.00}{262{,}144} = 32\text{k (256K) pages.}$$

[subtract powers 33-18=15; 2^{15}= 32,768 =~32k]

2. Question:

Given the set of specific address size and associated block size below. Calculate the number of page frames in the physical address space :

a) Physical address space 1 GB and the block [page] size of 64 KB; and

b) Given a block [page] size of 128 KB and a physical address space of 2 GB.

Answer:

2. Answer:

Given the set of specific address size and associated block size below. Calculate the number of page frames in the physical address space :

a) Physical address space 1 GB and the block [page] size of 64 KB;

If the virtual address space is 1 GB and the block size is 64 KB there are:

Number of Pages frames = $\frac{\text{Address space}}{\text{Block size}}$

$= \frac{1\ \text{GB}}{64\ \text{KB}} = \frac{2^{30}}{2^{16}} = \frac{1{,}073{,}741{,}824}{65{,}536}$ = 16k (64K) page frames.

[subtract powers 30-16=14; 2^{14}= 16,384 =~16k]

b) Given a block [page] size of 128 KB and a physical address space of 2 GB.

If the virtual address space is 2 GB and the block size is 128 KB there are:

Number of Pages frames = $\frac{\text{Address space}}{\text{Block size}}$

$= \frac{2\ \text{GB}}{128\ \text{KB}} = \frac{2^{31}}{2^{17}} = \frac{2{,}147{,}483{,}648}{131{,}072}$ = 16k (128K) page frames.

[subtract powers 31-17=14; 2^{14}= 16,384 =~16k]

3. Question:

Given the page table, on the right, and the set of virtual addresses (*p, o*) below. Answer the following questions for each of the virtual address:

a) Will the virtual address generate a page fault?

b) If it does, generate a page fault, what must the MMU do? [If not; what does it do then?]

c) What is the physical address; if 'no' to a)?

3.1) (*p, o*) = Ox 00020006; &

3.2) (*p, o*) = Ox 0006000F.

Given *p* is a 16-bit number and *o* is a 16-bit number.

Page table

(p)	*(i)*
…	…
07	00
06	XX
05	03
04	02
03	XX
02	01
01	XX
00	XX

Answer:

Note1: the page table only displays the lower 2 Hex digits, of the page number, not all four.
Note 2: XX denotes no page frame number allocated, yet!

3. Answer:

Given the page table, on the right, and the set of virtual addresses (p, o) below. Answer the following questions for each virtual address:

Page table

(p)	*(i)*
…	…
07	00
06	XX
05	03
04	02
03	XX
02	01
01	XX
00	XX

3.1) (p, o) = Ox 00020006

a) Will the virtual address generate a page fault?

- No.

b) If it does, generate a page fault, what must the MMU do? [If not; what does it do then?]

- No page fault; MMU just translates the virtual address to a physical address using the page table.

c) What is the physical address; if 'no' to a)?

- Ox 00020006 → Ox 00010006

3.2) (p, o) = Ox 0006000F

a) Will the virtual address generate a page fault?

- Yes.

b) If it does, generate a page fault, what must the MMU do? [If not; what does it do then?]

- Here a '*Page fault*' occurs. During the address translation the page was found not to be in the page table. This causes a trap to the *operating system, and a service routine loads the page [block] from disk [secondary memory].*

c) What is the physical address; if 'no' to a)?

- The algorithm that is used to allocate the page frame number; will allocate this address; at present it is not in the page table.

4. Question: exercise in conversion from page to page frame number

- A computer system operates using a paged virtual memory system (no segmentation). The processor address bus is 40 bits wide, the main memory in the system is 1GB and the page size is 128KB.
  a. What is the size of the virtual address space?
  b. How many pages are there in the virtual memory?
  c. How many page frames are there in the main memory?
  d. The 40-bit virtual address will be split into two bit fields, the *page number* and the *offset*, state how many bits there will be in each field.
  e. Suppose that part of the page table is given below (entries for 'page frame' number given in hexadecimal representation of a 23-bit binary value). Where it is possible to determine the physical address, state what this will be for the following addresses (given in hexadecimal):
     i. 0x000000ACEF
     ii. 0x00000B0020
     iii. 0x000004FCDA
  f. Now, given the Page table opposite; which of the 'page frame Number(s)' are Viable? If not why?

Virtual Page Number	Page Resident	Page Dirty	Page Frame Number
0	1	0	3AAC00
1	1	1	0456BFC
2	0	0	-
3	1	0	000200
4	0	0	-
5	1	1	01576F

4. Answer

a. What is the size of the virtual address space?

The virtual address space is the size of memory that could be addressed using the number of address bits provided. In this case we have 40 address bits, giving an address space of 2^{40} or 1 TB (T=terra, 1024 GB).

b. How many pages are there in the virtual memory?

Each page is 128 KB or 2^{17} bytes. The address space is 2^{40}. The number of pages is given by:

number of pages = address space / page size = $2^{40}/2^{17} = 2^{23}$ pages (8 M Pages)

c. How many page frames are there in the main memory?

The main memory size is 1GB or 2^{30}B. The number of page frames = memory size / page size = $2^{30}/2^{17} = 2^{13}$ pages (8 K Pages).

Note from b) and c) that the virtual address can reference over 8 million pages, of which less than 10,000 can be stored in the physical memory at any one time.

4. Answer

d. P*age number* and the *offset*, state how many bits there will be in each field?

There are 2^{23} pages to be addresses, this we need log2(2^{23}) bits to address them.

Therefore, 23 bits are required for the page [number] field.

Each page is 128 KB (2^{17} bytes), so log2(2^{17}) bits are needed to specify the byte address in the page.

Thus 17 bits are required for the offset.

Note that 23 bits for the page number and 17 bits for the offset leads to the 40 bits we require for the complete address.

page [number] field	offset
23 bits	17 bits

As 1GB or 2^{30}B page

Frame fields are:

page [frame] field	offset
13 bits	17 bits

4. Answer e.

i. Firstly split the address into the frame number and offset:

1) 000000ACEF hex

2) 0000 0000 0000 0000 0000 0000 1010 1100 1110 1111 binary

3) Split into page number (23 bits) and offset (17 bits)

page [number] field	offset
23 bits	17 bits

Virtual Page Number	Offset
000,0000,0000,0000,0000,0000	0,1010,1100,1110,1111

4) Virtual Page Number = 0, so look in page table entry for page 0

Virtual Page Number	Page Resident	Page Dirty	Page Frame Number
0	1	0	3AAC00

5) This shows that the page is loaded into memory in *page frame* $3AAC00_{16}$.

6) This value is used to replace the 23 bits of the virtual page number in the address so:

page [frame] field	offset
13 bits	17 bits

Page Frame Number	Offset
011,1010,1010,1100,0000,0000	0,1010,1100,1110,1111

This gives the address as

7) $011101010101100000000000101011001110111 1_2$

8) 0x755800ACEF

4. Answer e.

ii. Firstly split the address into the frame number and offset

1) 00000B0020 hex
2) 0000 0000 0000 0000 0000 1011 0000 0000 0010 0000 binary
3) Split into page number (23 bits) and offset (17 bits)

page [number] field	offset
23 bits	17 bits

Virtual Page Number	Offset
000,0000,0000,0000,0000,0101	1,0000,0000,0010,0000

4) Virtual Page Number = $101_2 = 5_{10}$, so look in page table entry for page 5;
5) This shows that the page is loaded into memory in page frame $01576F_{16}$.
6) This value is used to replace the 23 bits of the virtual page number in the address so:

page [frame] field	offset
13 bits	17 bits

Page Frame Number	Offset
000,0001,0101,0111,0110,1111	1,0000,0000,0010,0000

This gives the address as

7) $000000101010111011011111\ 0000000000100000_2$
8) 0x02AEDF0020

4. Answer e.

iii. Firstly split the address into the frame number and offset

- 000004FCDA hex
- 0000 0000 0000 0000 0000 0100 1111 1100 1101 1010 binary
- Split into page number (23 bits) and offset (17 bits)

page [number] field	offset
23 bits	17 bits

Virtual Page Number	Offset
00000000000000000000010	01111110011011010

- Virtual Page Number = $10_2 = 2_{10}$, so look in page table entry for page 2.
- This shows that the page is **not** resident. So a page fault must be generated and the operating system must load page 2 into one of the page frames and update the page table. The physical address can only be determined when this process has been completed.

4. Answer f.

f. Now, given the Page table; which of the 'page frame Number(s)' are Viable? If not why

Page Frame Number Base10	Page Frame Number Base16	Viable	Greater than 0x2000 (8192 base 10)
3845120	3AAC00	no	yes
4549628	0456BFC	no	yes
	-	-	-
512	200	yes	no
	-	-	-
87919	01576F	no	yes

From, answer to 4.c. there are :-
Base10 Base16
8192 2000 possible page frames that are viable; depicted in the table above; although this was just a "**exercise in conversion from page to page frame number**" it is worth noting that the OS and the algoritmics aligned to the page table in the OS would not facilitate page frames + offset addresses greater than 30 bits, in this case, as this is the addressable space in the physical memory.
Reason ['no' in viable column] for not being 'viable' is that their "frames + offset addresses" are greater than 30 bits.

Questions

5. Answer:

Explain the difference between a virtual memory address and a physical memory address.

- A physical memory address is an address which corresponds to a fixed location in physical RAM memory.
- A virtual address is an address in a conceptual memory space which a process (program) thinks it is operating.
- Virtual addresses are normally translated to physical addresses by Memory Management Unit (MMU) hardware.

Questions

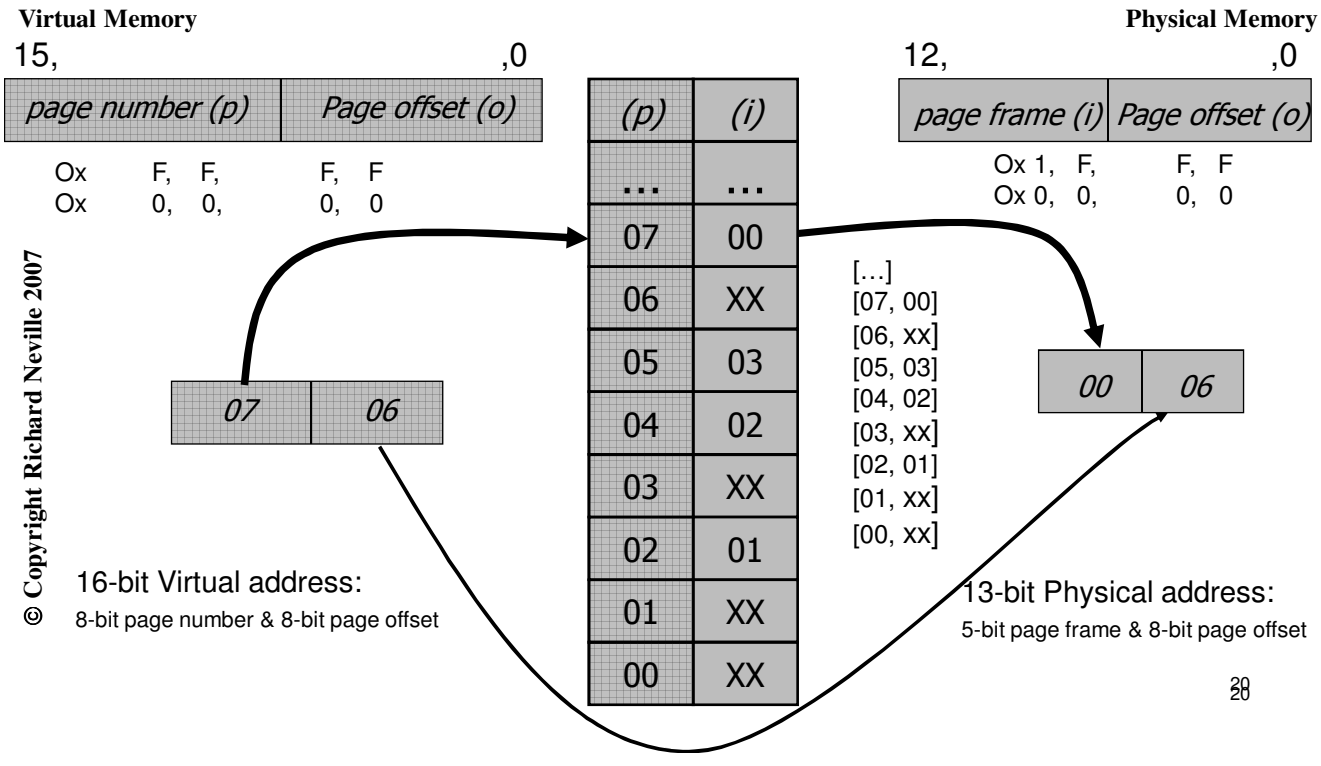
6. A 16 bit [virtual] address is divided into an 8-bit page number and an 8-bit page offset; whereas the associated physical address a 7-bit page frame with the appropriate page offset address sizing. Show, with the aid of a diagram, the table structure necessary to convert this virtual address into a real [physical] address.

Answer:

Questions

6. A 16 bit [virtual] address is divided into an 8-bit page number and an 8-bit page offset; whereas the associated physical address a 5-bit page frame with the appropriate page offset address sizing. Show, with the aid of a diagram, the table structure necessary to convert this virtual address into a real [physical] address.

Answer:



7. What is a 'page fault'? Describe how a page fault is handled by the Memory Management Unit and the operating system..

Answer:

In a paged virtual memory system, pages may have copies in real memory or may exist only on backing storage (usually disk or secondary memory).

If the MMU attempts an address translation but the page "**table indicates that there is no copy in real memory, this is a page fault. The MMU interrupts the CPU and traps to an OS page fault handler.**

This is responsible for both selecting a page to reject from real memory – and writing it to disk if it has changed since it was last read" – and organising the read of the required page from disk to real memory.

It then updates the page tables and transfers control back to the original program to re-execute the instruction which caused the fault.

Revision Exercises

- Scan read Lecture 13's Questions.
  - Answer Lecture 13's Questions
    - Particularly those questions you had difficulties with when you first tried them.