Virtual memory
Solved exercises

Exercise 1. Be a 20-bit computer with paged virtual memory with 1 KB pages and a total physical memory of 256 KB. It is requested, in a reasoned and brief way:

a) What is the format of the virtual address? Enter the fields and the number of bits in them.
b) What is the maximum number of entries in the page table (of one level)?
c) How many page frames does the main memory have?
d) What are the fields that are included in a page table entry? Also indicate what each of the fields is used for.

Solution:

a) The pages occupy 1 KB = 2^{10} bytes. As the virtual address occupies 20 bits, 20 -10 = 10 bits are used for the page number. Therefore, the format uses the top 10 bits of the address to represent the page number and the bottom 10 bits to represent the offset within the page.
b) The maximum number of entries in the page table matches the maximum number of pages, i.e. 2^{10} = 1024 entries.
c) The number of page frames is given by 256 KB / 1 KB = 256 frames.
d) Each entry in the page table includes, but is not referred to:
    a. Presence bit
    b. Modified bit
    c. Validity bit
    d. Permission bits
    e. The field in which the frame is stored.

Exercise 2. Consider a 32-bit computer that has a virtual memory system that uses 16 KB pages and has a 1 GB main memory installed. Indicate in a reasoned manner:

a) The format of the virtual address.
b) The maximum number of pages on this computer.
c) The number of page frames on this computer.
d) The size of the block that is transferred between disk and main memory when a page failure occurs
e) The element of the computer that generates the page failure and who treats it.

Solution:

a)

Id. página	Desplazamiento
32 – 14	2^{14} = 16 KB
18 bits	14 bits

b) $2^{32} / 2^{14} = 2^{32-14} = 2^{18}$ **frames**

c) $2^{30} / 2^{14} = 2^{30-14} = 2^{16}$ **pages**

d) The size of a page is **16 KB**

e) The **MMU throws the exception** and the operating system **page failure routine** handle the exception

Exercise 3. A computer has a virtual memory system implemented by paging that uses 8 KB pages. The computer provides a virtual memory space of 2^{32} bytes and has 2^{23} bytes of physical memory. If the page table for a running program is as follows:

Presence bit	Modified bit	Page Frame/ Swap Block
1	0	1
0	0	7
1	1	9
1	0	14
1	0	8
1	1	3
0	0	25
0	1	16
0	0	23
1	0	78

It is requested:

a) Enter the format of the virtual address.
b) Enter the physical address corresponding to the virtual address 0x0000608A.
c) What is the size of the virtual address space of this program?
d) Express the size of the main memory in MB.

Solution:

a) The computer has pages of 8 KB = 2^{13} bytes. Because the virtual memory is 2^{32} bytes, the top 19 bits of the address are used for the page and the bottom 13 bits for scrolling within the page.
b) The address 0x0000608A == 0000 0000 0000 0000 0110 0000 1000 1010
The top 19 bits are 0000 0000 0000 0000 011 = 3. The address refers to page 3, which is in memory, in the frame 14 = 1110
The physical address is 0000 0000 0000 0001 1100 0000 1000 1010 == 0x 0001C08A
c) This program takes up 10 pages, then the address space it occupies is 10 x 8 = 80 KB.
d) The main memory has 2^{23} bytes = 2^{13} KB = 2^3 MB = 8 MB.

Exercise 4. A computer (which addresses memory by bytes) is available with a virtual memory system that uses 16-bit virtual addresses and 2 KB pages. The computer has an installed physical memory of 8 KB. It is requested:

a) What is the maximum size, in KB, of the virtual memory that can be addressed?
b) Indicate the maximum number of pages that a program running on this computer can have.
c) Indicate the format of the virtual address used on this computer.
d) Specify the size of the page frame.
e) Enter the number of page frames in the physical memory.
f) Indicate the format of the physical address of this computer.
g) What is the maximum number of entries that the page table associated with a program running on this computer can have, assuming that it is a single-level page table?
h) Enter at least two fields from each entry in the page table and say what they are used for.

Solution:

a) 2^{16} memory addresses can be addressed. As the computer addresses the memory per byte, one byte is stored in each address and therefore the size of the virtual memory will be 64KB.
b) As each page is 2KB, at most there will be 64KB/2KB = 32 pages.
c) The format of the virtual address is as follows:

Page number = 5 bits	**Offset = 11 bits**

d) The size of the page frame is equal to the size of the page, that is, 2 KB.
e) If the physical memory is 8KB. There will be 8 KB/2 KB = 4 frames.

f) The format of the physical address is as follows:

Frame No. = 2 Bit	Offset = 11 bits

g) Since there are 32 pages and the page table is single-level, you will have at most 32 entries: one for each possible page of virtual memory.

h) Possible fields:
P/A = Present/Absent, indicates whether a page is in Main Memory or not.
M = Modified, indicates whether the page in Main Memory has been modified or not, so that if it has been modified when a page failure occurs before bringing a new page to Main Memory it will have to be recorded on the storage device used to support the virtual memory.
Frame Number = indicates the frame in which a page is located if it is present in Main Memory

Exercise 5. A computer has a virtual memory system implemented by paging that uses 4 Kbyte pages. The computer provides a virtual memory space of 2^{32} bytes and has 2^{18} bytes of physical memory. If the page table for a running program is as follows:

Presence bit	Modified bit	Page Frame/ Swap Block
1	0	1
0	0	8
1	1	9
1	0	14
1	0	5
1	0	7
0	0	25
0	1	16

It is requested:
a) Enter the format of the virtual address.
b) Enter the physical address corresponding to the virtual address 0x00005B83
c) What is the size of the virtual address space of this program?

Solution:

a) The virtual addresses are 32-bit, 12 bits are used for the frame and 20 for the page number.

b) Dividing the virtual (logical) address by the page size gets the page number, and the reminder of the division is the offset within the page.

23456/4096=5 and reminder=2976

If we look at the table of pages of the process, page 5 is in frame 4, the physical address is obtained as:

(Frame No. x Page Size) + Offset = (4 x 4096) + 2976 = 19360

c) This program occupies 8 pages of 4 KB, that is, it occupies 32 KB.

Exercise 6. Given a system that employs a paging scheme with the following characteristics:
- The number of pages per process is a maximum of 8096 (8K).
- The page size is 32KB.
- The physical memory is 16 MB.
- The disk access time is 120 ms, 50 ns memory and 10 ns to TLB.

a) What is the format of virtual addresses? Specifies the size of the fields and their meaning. It also indicates the size of the addressable space virtually.
b) What is TLB? What is its function?
c) Based on the content of the TLB and the page table of any given process, indicate the accessed memory address for the following virtual addresses (expressed in hexadecimal):

001A007
0007100
00140C2

Note that the entries in the page table have 3 control bits, of which the two most significant correspond to the presence bit and the modification bit respectively.

TLB

2	A1C
5	C04
1	043

Tabla de paginas

0	FF1
1	043
2	A1C
3	203
4	76A
5	C04
6	60F
7	663
8	011

d) Attending to the pages accessed in the previous section, it indicates if writing to disk should occur in case they are replaced in memory. Why?.
e) Reasonably obtain the access time for each of the virtual addresses in section "c".

Solution:

a) Num. pages = 8096 = 2^13 → 13-bit
Page size = 32 KB = 2^15 →15-bit

Num. Frames = 16 MB / 32 KB = 2^24/2^15=2^9 → 9 bits
Meaning Fields: Page Number + Offset
Virtually addressable space → 2^28 = 256 MB

b) Translation Lookaside Buffer. It is an associative cache that stores the page frames where the most recently referenced pages are stored. Its function is to reduce the search time of the real address from a virtual address, saving access to the page table.

c) 001A007 → 0000 0000 0001 1010 0000 0000 0111

Scroll. = 010 0000 0000 0111
Num. Page = 0000 0000 0001 1 (3)
Not in TLB → 203 0010 0000 0011

Bit Presence = 0
Bit Modif. = 0

NOT IN MAIN MEMORY

0007100 → 0000 0000 0000 0111 0001 0000 0000

Scroll. = 111 0001 0000 0000
Num. Page = 0000 0000 0000 0 (0)

Not in TLB, FF1 → 1111 1111 0001

Bit Presence = 1
Bit Modif. = 1

Num. Frame 1 1111 0001

00140C2 → 0000 0000 0001 0100 0000 1100 0010

Scroll. = 100 0000 1100 0010
Num. Page = 0000 0000 0001 0 (2)

In TLB, A1C → 1010 0001 1100

Bit Presence = 1
Bit Modif. = 0

Num. Frame 0 0001 1100

d) 001A007 → Not in main memory.

0007100 → Modification bit 1. The page on disk must be refreshed.

00140C2 → Modification bit 0. The page has not been modified.

e) 001A007 →TLB (10 ns)+Pages Table(50ns)+Disk(120ms)

0007100 → Table Pages (50ns) + Memory (50ns)

00140C2 → TLB (10ns) + Memory (50ns)

Exercise 7. Consider a computer with the following characteristics:

- 32-bit CPU, with 36-bit virtual addresses (ability to address 2^{36} bytes). 32-bit address and data buses.
- Paged virtual memory, with the following characteristics:
  - Three levels of page tables. Pages with storage capacity for 8KB
  - Each entry at any table level occupies one word.
  - The first-level and second-level page tables are the same size.
  - The size of each third-level page table is 1 page
  - TLB access time of 8ns.
- Main memory with access time of 70ns.

Consider that the TLB hit table is 90%.

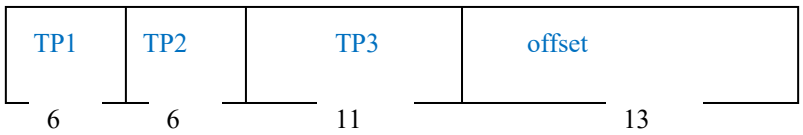
a) Describe, for the specified system, the fields into which an address can be decomposed
b) Calculate the average memory access time of a piece of data (for reading or writing) considering only cases where there is no page failure

Solution:

a) A virtual address (36-bit) consists of:
   - Offset within the page
   - Number of entries in the 3rd level page table (TP3)
   - Number of entries in the 2nd level page table (TP2)
   - Number of entries in the 1st level page table (TP1)

   The number of bits used in each of the fields is as follows:
   - Because a page consists of 2^{13} bytes, 13 bits are needed to determine the *offset* within the page.
   - Each entry in the 3rd level page table occupies a word, i.e. 2^2 bytes and each *TP3* occupies a page, i.e. 2^{13} bytes so there will be $2^{13} \div 2^2 = 2^{11}$ entries. Therefore, 11 bits are needed.

- The first and second level page tables (TP1 and TP2) are the same size, so the same number of bits will be used to select an entry in each of them. Of the 36 bits that consist of a virtual address, 24 (13 + 11) are used to search in TP3 and obtain the offset within the page, so the remaining 12 will be used to determine the TP1 and TP2 entries. 6 bits will therefore be used for each of these fields:



b) Average access time to a data in Main Memory:
- If there is success in TLB (probability: 0.9), the access is completed in 8ns: 0.9 x 8ns = 7.2ns
- If there is failure in TLB (probability: 0.1)
  i. The Main Memory is accessed to read the TP1: 70ns entry
  ii. The Main Memory is accessed to read the TP2: 70ns entry
  iii. The Main Memory is accessed to read the TP3: 70ns entry
  0.1 x (70+70+70) = 21ns
- The main memory is accessed to read (or write) the data: 70ns

Time = 7.2ns + 21ns + 70ns = 98.2ns

Exercise 8. Be a computer with the following characteristics:
- 32-bit word width
- 32-bit addresses
- 64 MB physical memory
- Virtual memory:
  - Page size: 2 KB
  - Replacement Policy: FIFO

At a certain point, a process is running on this computer that is assigned exclusively 8 page frames. The code snippet for the period considered is:

```
s = 0;
m = 0;
r = 0;
for (i = 0; i < 2047; i++)
      s = s + a[i];
for (i = 0; i < 2047; i++)
      m = m + a[i] * b[i];
for (i = 0; i < 2047; i++)
      r = r + a[i] + c[i];
```

The loop control variable (i) and the variables s, m, and r are mapped to processor registers. All variables that appear are 32-bit integers. The structures a[i], b[i] and c[i] are assigned by the processor to consecutive addresses, in that order, starting from the 0x0000 address.

Considering only access to data addresses, it is requested:

a) Write the trace of the program, indicating the sequence of pages that are accessed.
b) Indicate the page misses that occur during execution.
(c) Repeat (b) assuming that the replacement policy is LRU.

Solution:

Each of the three *structures* a, *b* and *c,* will occupy a total of 2Kwords of 4 bytes. Consequently, taking into account that the page size is 2 Kbytes each of them will occupy 4 pages in the virtual memory system, 12 pages the three structures. Each page will contain 512 consecutive components of the vector.

On the other hand, the process is only assigned 8 page frames, so page misses will necessarily occur during code execution.

(a) the structure *shall* occupy the first 4 pages, P0, P1, P2 and P3; *b*, the following 4, P4, P5, P6 and P7; and *c* the remaining, i.e. P8, P9, P10 and P11.

The trace that is requested would be the succession of pages that are accessed during the execution:

First loop: access only to the elements a[i]:

{P0} [512 times], {P1} [512 times], {P2} [512 times], {P3} [512 times]

Second loop: Alternate access in each iteration to elements a[i] and b[i]:

{P0, P4} [512 times], {P1, P5} [512 times], {P2, P6} [512 times], {P3, P7} [512 times]

Third loop: Alternate access in each iteration to elements a[i] and c[i]:

{P0, P8} [512 times], {P1, P9} [512 times], {P2, P10} [512 times], {P3, P11} [512 times]

b) To determine the page failures that occur, we start from the assumption that none of the pages is resident at the initial moment of the execution of the code.

- First loop: the first access to the four pages P0, P1, P2 and P3 will fail: 4 failures
- Second loop: accesses to P0, P1, P2 and P3 will not fail; yes, the first accesses to the pages corresponding to b, P4, P5, P6 and P7: 4 failures. With these 8 pages, the available frames will have been completed.
- Third loop: the first access to P0 does not fail; the first access to each of the pages of structure *c*, yes: as there are no frames available the replacement policy will have to be applied, FIFO in this case: the pages to be replaced will be those corresponding to structure a, since they are the ones that have been resident for the longest time. Consequently, the second accesses to P0, P1, P2 and P3, will also produce failure: 8 misses

In total, therefore, with the FIFO replacement policy there are (4 + 4 + 8) misses = 16 misses

c) If the replacement policy followed is LRU *(Least Recently Used)* the pages that have passed the most time without being accessed will always be replaced. In our particular case, replacements are only needed in the third of the loops.

In this case, the pages corresponding to structure *a* would not be replaced, since they are reused in the loop itself, but those that correspond to structure *b.* As a consequence, the number of misses that occur is due only to the first accesses of *c,* while the accesses to pages of a will always find the corresponding page in memory. Therefore, the number of misses that occur is (4 + 4 + 4) = 12 misses, compared to the 16 that occurred with the FIFO policy.