Memory Management

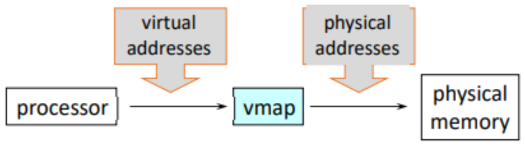
اللَّهُمَّ صَلِّ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّدٍ، كَمَا صَلَّيْتَ عَلَى إِبْرَاهِيمَ، وَبَارِكْ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّد،
كَمَا بَارَكْتَ عَلَى آلِ إِبْرَاهِيمَ، فِي الْعَالَمِينَ، إِنَّكَ حَمِيدٌ مَجِيدٌ.

By : Mohamed Gamal Maklad

- **Why to make memory management:**
  - To **provide a *convenient abstraction*** for programming **بتريح المبرمج من انه يخصص مكان تخزي لكل عمليه**
  - To **allocate memory resources** among competing processes to **maximize performance** with **minimal overhead**
- **Physical and virtual addressing Techniques**: **partitioning**, **paging**, **segmentation**
- **Why using virtual memory (VM)** : **باخد جزء من الهارد ديسك واستخدمه كميموري معايا**
  - **Enables a program to execute without need for complete data** in physical memory(Ram)
    - A **program can run on a machine with less memory** than it "needs"
    - **Many programs do not need all of their code and data at once** (or ever) – no need to allocate memory for it
  - **Processes cannot see the memory of others** **مفيش عمليه تنفع تشوف موارد عمليه تانيه**
  - OS will adjust(تعدل) amount of memory allocated to a process **based upon its behavior**

  Note➔ VM requires **hardware support** and **OS management** algorithms to pull it off
- **Old Memory Management** :
  - **Programs use physical addresses directly** And **OS loads job, runs it, unloads it**
  - **multiple processes** in memory want to ➔ **Overlap(تداخل) I/O and CPU of multiple jobs**
  - Can do it a number of ways By **Fixed and variable partitioning, paging, segmentation**
  - **There is a Requirements for multiprogramming :**
    - ***Need protection*** restrict which addresses jobs can use **بتحمي أجزاء كل عمليه من العمليات التانيه**
    - ***Fast translation*** **lookups need to be fast**

      هنا معناه انه بيحول العنوان اللي المبرمج شايفه الي العنوان اللي يقدر physical memeory انه يتعامل معاه
    - ***Fast change*** **updating memory hardware on context switch** **اقدر ابدل بين البرامج و بعض بسرعه**
- **Virtual Addresses**:
  - **use *virtual addresses* make it easier to manage the memory of processes**
  - Virtual addresses are independent of the actual physical location of the data referenced
  - **Instructions executed by the CPU** issue virtual addresses
  - Virtual addresses are **translated by hardware** into **physical addresses** (with help from OS)
  - ***virtual address space*** The **set of virtual addresses that can be used by a process comprises**

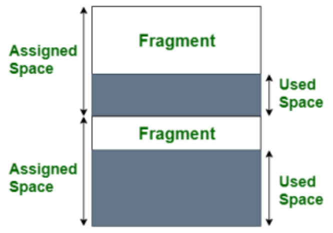


ال processor هنا بيجيله Virtual address يقوم يبعته لحاجة اسمها virtual map دي بيتقوم تحوله الي physical address وبعدين تروح بقي ال physical memory تجيب محتوياته منها

اللَّهُمَّ صَلِّ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّدٍ، كَمَا صَلَّيْتَ عَلَى آلِ إِبْرَاهِيمَ، وَبَارِكْ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّدٍ، كَمَا بَارَكْتَ عَلَى آلِ إِبْرَاهِيمَ، فِي الْعَالَمِينَ، إِنَّكَ حَمِيدٌ مَجِيدٌ.

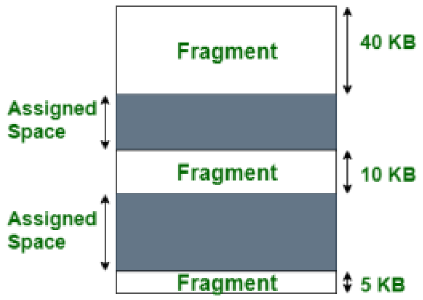
❖ Internal Fragmentation:

خلاصة الحته دي ان كل process بخصص ليها جزء معين بس هي مثلا مش هتستخدم الجزء ده كله مش هتحتاجه فهيفيض منها مساحه ف الجزء اللي هتستخدمه هنسميه Used والجزي الفاضل هنسميه fragment و internal ليه علشان الجزء اللي فاضل ده أصلا جزء داخلي مخصص ليها



❖ External Fragmentation:

هنا بقي هو مخصص جزئ لكل عمليه بس بين كل جزء و التاني حاطط مسافه فاضيه فلو مثلا_ عندي process محتاجه جزء اكبر من اللي متخصص ده هيؤدي الي ان هي تتقسم



- ***External fragmentation***: **can be solved by re-mapping between VA** (Virtual Address) **and PA** (Physical Address)
- ***Internal fragmentation***: **can be solved if the page size is relatively small**

❖ Fixed Partitions: **(بتحدد جزء معين لكل عمليه)**

- ***Hardware requirements***: **base register**
- **Physical address = virtual address + base register**
- **Base register loaded by OS when it switches to a process**
- **Size of each partition is the same and fixed**
- ***Advantages*** ➔ **Easy to implement, fast context switch**
- ***Problems*** ➔ **Internal fragmentation** : **memory in a partition not used by a process is not available to other processes** هتعمل حجم ثابت ده هيعمل مشكلة لو مثلا عمليه صغيرهومش هتحتاج الحجم ده كله زي ما قولنا فوق
  - ***Partition size:*** one size does not fit all (فكرة انك تعمل حجم ثابت مش هيناسب كل العمليات)

هنا بقي بخصص حجم ثابت لكل عمليه وبحدد مكان كل عمليه عن طريق physical address اللي مكون من ايه مثلا لو قولتلك قولي عنوانك هتقولي انا ساكن في عماره كذا اللي هو base register طب يعم في الدور الكام هتقولي التاني مثلا اللي هو يمثل هنا virtual address

❖ Variable Partitions:

- ***Hardware requirements***: ***base register* and *limit register***
- **Physical address = virtual address + base register**
- ***If (physical address > base + limit)*** then exception fault (Will Raise Error that is Called fault)
- ***Advantages*** ➔ **No internal fragmentation**: allocate just enough for process
- ***Problems*** ➔ **External fragmentation** : job loading and unloading produces **empty holes scattered throughout memory**

في هنا مشكله لو في عمليه خلصت ودخل مكانها عمليه تانيه اقل منها ف الحجم ف كده هيكون في فراغ بين العمليات فعلشان نحل المشكله دي هنستخدم حجا اسمها Compaction

❖ Compaction :

- ***Compact memory by copying***
  - **Swap a program out** •**Re-load it, adjacent to another** •**Adjust its base register**

هنا بيروح اول حاجه يعمل suspended للعمليات يعني يوقفها لحد ما يخلص وبعد كده يجمع العمليات تحت بعض و بعد كده يروح يحط الفراغات دي تحت خالص

Note➔ Processes must be <u>suspended</u> during compaction

بنقسم الميموري الي أجزاء متساويه تمام والاجزاء دي بتكون يا اما page او frame ال frame هو اسم ل block في memory اما page هو اسم (virtual storage block ولازم حجم page = frame كل page بيقابلها frame

❖ **<u>Modern technique: Paging</u>** : (هنا بنقسم البرنامج الي أجزاء ➔page)

➢ ***Solve the external fragmentation*** problem by using **fixed sized units in both physical and virtual memory**

(Size of Virtual address == physical address)

➢ ***Solve the internal*** fragmentation problem **by making the units small**

❖ **<u>Benefits of Paging</u>** :

➢ **<u>For the programmer</u>**

- **Processes view memory as a contiguous address space** from bytes 0 through N – a ***virtual address*** space
- **N is independent of the actual hardware**

لازم يكون عندنا طريقه نحول من Virtual-to-physical mapping بدون ما البرنامج بتاعنا يتدخل

➢ **<u>For the memory manager</u>**

- **<u>Efficient use of memory, because very little internal fragmentation</u>**
- **No external fragmentation at all**
  - ➢ **No need to copy big chunks of memory around to coalesce free space**

يفضل ان حجم page يكون دايما مرفوع ل أس 2

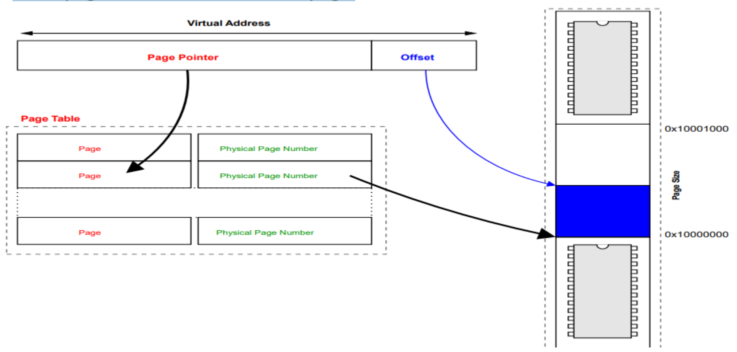
Page size is always a power of 2➔ Examples: 4096 bytes = 4KB, 8192 bytes = 8KB

Reason: Multiplication or division by 2 ^{x} can be replaced by bitwise **left shift or bitwise right shift**.

They are extremely fast, in **comparison to regular multiplication or division by arbitrary integer**

❖ **<u>Virtual Address</u>**:

➢ **A virtual address consists of two parts** the <u>page and an offset into that page</u>.



اللَّهُمَّ صَلِّ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّدٍ، كَمَا صَلَّيْتَ عَلَى إِبْرَاهِيمَ، وَبَارِكْ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّدٍ، كَمَا بَارَكْتَ عَلَى آلِ إِبْرَاهِيمَ، فِي الْعَالَمِينَ، إِنَّكَ حَمِيدٌ مَجِيدٌ

- **Address translation** :
  - **Translating virtual addresses**
    - a virtual address has two parts: **virtual page number & offset**
    - virtual page number (**VPN**) **is index into a page table**
    - page table entry contains **page frame number** (PFN)
    - **physical address** is PFN+offset
  - **Page tables**
    - **map virtual page number (VPN) to page frame number (PFN)**
    - **VPN is simply an index into the page table**
- **Page Fault (Error)**: Since the page-tables are under the control of the operating system, **if the virtual-address doesn't exist in the page-table** then the operating system knows the process is trying to **access memory that has not been allocated to it and the access will not be allowed.**

  ده بيحصل لما يكون معاك عنوان مش موجود أصلا في page table ف انت مش هتعرف accessعليه
- **Page Table Entries (PTEs)**:

1	1	1	2	20 (determined by the size of physical memory)
M	R	V	Prot	Page Frame Number

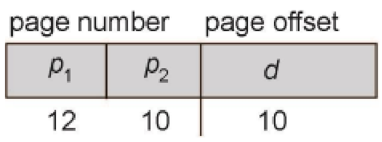
  - ***The(M) Modify*** bit says **whether or not the page has been written**
    - It is **set when a write to the page occurs** بتعرفنا هل تم الكتابه علي البيدج ده ولا لأ
  - ***The (R) Reference*** bit **says whether the page has been accessed**
    - **It is set when a *read* or write to the page occurs** هل تم القرأه او التعديل علي البيدج دي
  - ***The(V) Valid*** **bit says whether or not the PTE can be used**
    - **It is checked each time the virtual address is used** هل هقدر استخدم المعلومات اللي فيها ولا لازم تتحدث
  - ***The (Prot) Protection*** bits **say what operations are allowed on page**
    - Read, write, execute
  - **The page frame number (PFN) determines physical page** الفريم المقابل للبيدج
- ***Single level page table size*** is too large → **4KB page, 32 bit virtual address, 1M entries per page table**
- **Hierarchical Page Tables**:
  - Break up the **logical address space into multiple page tables**
  - A simple technique is a two-level page table
  - **We then page the page table**(هنقسم الجدول الي جداول وكل جدول فيه جداول هو مسؤال عنها)
- **Two-Level Page Tables**:
  - **Virtual addresses (VAs) have three parts**:

    Master page number, secondary page number, and offset
  - ***Steps***:
    - ***Master page*** table maps VAs to secondary page table
    - ***Secondary page*** **table maps page number to physical page**

- ***Offset indicates*** where in physical page address is located

❖ **32 bit Two-Level Paging Example** :

- A logical address (on 32-bit machine with 1K page size) is
  - a **page number** consisting of **22 bits**
  - a **page offset** consisting of 10 bits ➔ (1k page size $=2^{10}$)
- Since the page table is paged, the page number is further
  - a 12-bit page number
  - a 10-bit page offset➔ (1k page size $=2^{10}$)



❖ **Paging Advantages** :

- ***Easy to allocate memory***
  - **Memory comes from a free list of fixed size chunks**
  - **Allocating a page is just removing it from the list**
  - **External fragmentation not a problem**
- ***Easy to swap out chunks of a program***
  - All **chunks are the same size**
  - **Use valid bit to detect references to swapped pages**
  - **Pages are a convenient multiple of the disk block size**

❖ **Paging Limitations** :

- **•Can still have internal fragmentation**
  - **Process may not use memory in multiples of a page**
- ***Memory reference overhead***
  - **2 references per address lookup (page table, then memory)(ده بيؤدي اني اخسر وقت و مساحه)**
  - **Even more for two-level page tables!**
  - ***Solution* use a hardware cache of lookups لعمل نسخه مؤقته و سريعه للحاجات اللي بستخدمها كتير**

❖ **What if a process requires more memory than physical memory**?

- ***Swapping*** ➔ Move one/several/all pages of a process to disk
- The **freed physical memory can be mapped to other pages**
- **Processes that use large memory can be swapped out (and later back in)**

❖ **A variation of paging: Segmentation** :

- ***Segmentation*** is a technique that partitions memory into logically related data units
  - **Module, procedure, stack, data, file, etc**.
  - Virtual addresses become **<segment #, offset>**
  - **Units of memory from user's perspective**
- **Natural extension of variable-sized partitions**
  - In **Variable-sized partitions** = 1 segment/process

هنا بيعتمد انه يقسم كل جزء علي حسب الداتا اللي موجوده فيه التقسيم ده هنسميه segment علشان أوصل ل segment هحتاج رقم segment و offset

- but **Segmentatio**n = many segments/process

➢ ***Paging***

- **view an address space as a linear array of bytes**
- **divide it into pages of equal size** (e.g., 4KB)
- **use a page table to map virtual pages to physical page frames**
- **page (logical) => page frame (physical)**

جه في بالك سؤال صح هتقولي طب ما **paging** بيقسم برضه هقولك سؤال حلو هو **paging** بيقسم بس تقسم كده و خلاص هقسم الجزء ده و خلاص اما **segmentation** بيقسم معتمد ان كل جزء له نفس البيانات يعني نظام بيقسم المتشابه مع بعضه اما **paging** بيقسم وخلاص مش لازم يكونوا متشابيهين

❖ Segmentation Advantages :

➢ absent segmentation, a linker **takes a bunch of independent modules** that call each other and linearizes them that Cause **More Logical**

➢ **Facilitates sharing and reuse** a segment is a **natural unit of sharing** – a subroutine or function

➢ A **natural extension of variable-sized partitions**

- ***variable-sized partition*** = ***1 segment/process***
- ***segmentation*** = ***many segments/process***

❖ Hardware support :

➢ Segment table

▪ **multiple base/limit pairs, one per segment**

▪ **segments named by segment #, used as index into table**

- a virtual address is <segment #, offset>

▪ **offset of virtual address added to base address of segment to yield physical address**

❖ Segmentation with Paging Combining segmentation and paging :

➢ **modern architectures** support both segments and paging

➢ Use segments **to manage logical units**

• **segments vary in size, but are typically large** (multiple pages)

➢ Use pages to **partition segments into fixed-size chunks**

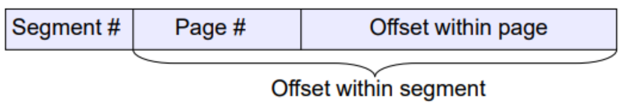
• each segment **has its own page table**

• **there is a page table per segment, rather than per user address space**

➢ **memory allocation becomes easy once again**

• **no contiguous allocation, no external fragmentation**

نشرح دي بمثال يعني عندنا الكليه مقسمانا علي حسب الحروف ف ده تقسم ب **segmentation** اهو معتمد علي الحروف يجي بقي يلاقوا اسم محمد منتشر اوي يقولك لا احنا هنقسمهم كمان فبيعمل **paging** ليهم و يقسمهم اكتر علشان لقي انهم كتير ف كده علشان اوثل لحاجه هحتاج رقم **segment** و رقم **page** و رقم **offset** بتاع البيدج يعني عاوز أوصل ل اسم محمد جمال هحتاج رقم **segment** هقولك 5 مثلا هتقولي في كتير هقولك روح سكشن 3 مثلا اللي هو **page number** هتقولي تمام عندي كذا محمد جمال هقولك هو بالتحديد رقم 11 اللي هو هنا رقم **offset**



اللَّهُمَّ صَلِّ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّدٍ، كَمَا صَلَّيْتَ عَلَى إِبْرَاهِيمَ، وَبَارِكْ عَلَى مُحَمَّدٍ وَعَلَى آلِ مُحَمَّد، كَمَا بَارَكْتَ عَلَى آلِ إِبْرَاهِيمَ، فِي الْعَالَمِينَ، إِنَّكَ حَمِيدٌ مَجِيد

By: Mohamed Gamal Maklad