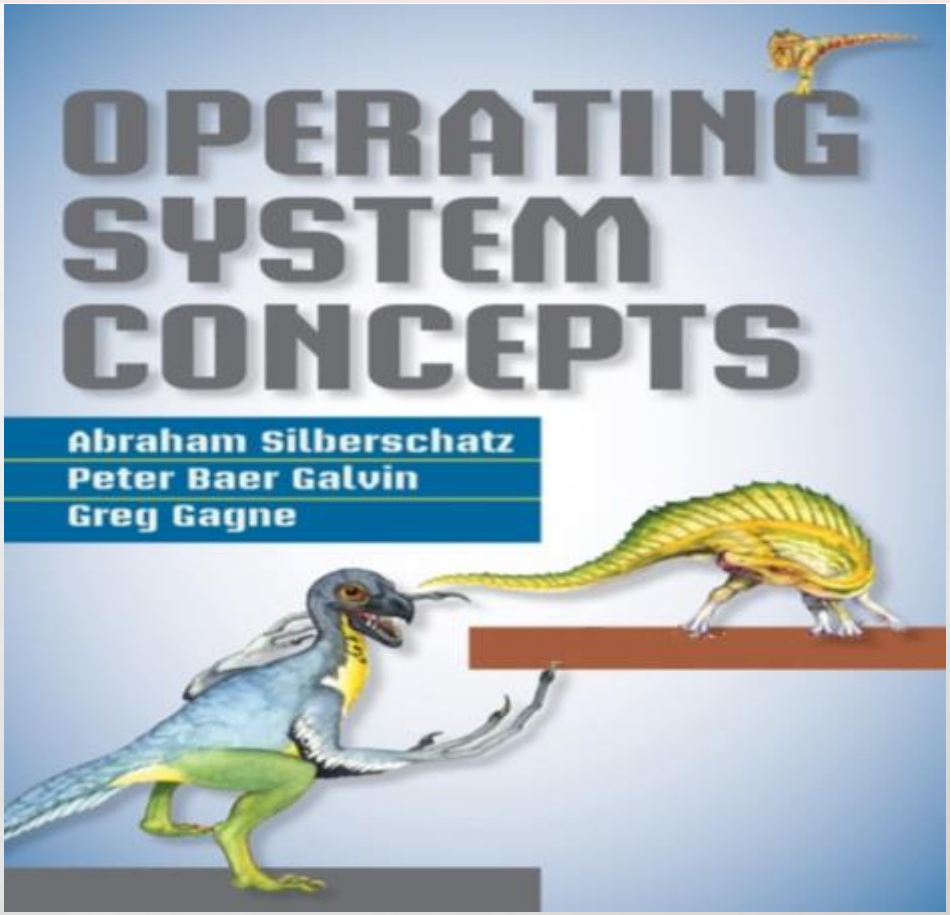
Mustansiriayah University
Collage of Education
Computers Science Department

Chapter Two
Part 2



Fourth Class

Dr. Hesham Adnan ALABBASI

2019-2020

2.4 Hardware Protection

- To improve system utilization, the O.S began to share system resources among several programs simultaneously.

لتحسين استخدام النظام، بدأ O.S في مشاركة موارد النظام بين العديد من البرامج في وقت واحد.

- Multi programming put several programs in memory at the same time. This sharing created both **improved utilization** and **increased problems**.

البرمجة المتعددة تضع العديد من البرامج في الذاكرة بنفس الوقت .وقد أدى هذا الشيء إلى تحسين الاستخدام ولكن زيادة المشاكل.

- When the system was run **without** sharing an error in a program could cause problems for only the one program that was running. **With** sharing many processes could be affected by a bug in one program.

عندما يعمل نظام التشغيل من دون المشاركة فأن الخطأ في برنامج يمكن أن يسبب مشاكل فقط في نفس البرامج الذي كان قيد التشغيل .مع المشاركة , العديد من ال processes يمكن أن تتأثر بخطأ في برنامج واحد.

2.4.1 Dual Mode Operation

- To ensure proper (مناسب أو لائق) operation, we must protect the O.S and all programs and their data from any malfunctioning (خلل) program.

لضمان التشغيل السليم يجب حماية O.S. وجميع البرامج والبيانات الخاصة بهم من أي خلل في البرنامج.

- Protection is needed for any shared resource. The H/W support to differentiating among various modes of executions. Therefore we need two separate modes of operation:

نحتاج هذه في حماية أي مورد مشترك .ال H/W صممت لكي تدعم التمييز بين مختلف أساليب التنفيذ .لذلك يتم فصل التشغيل الى نموذجين :

1- **User mode**

2- **Monitor mode** (also called **kernel mode, system mode, or privileged mode**).

- A bit called **mode bit** is added to H/W to indicate (تشير) the current mode;

- **Monitor (0):** execution is done on behalf of the O.S يتم التنفيذ نيابة عن
- **User (1):** execution is done on behalf of the USER يتم التنفيذ نيابة عن

This protect the O.S from errant (المخطئين) users and errant users from one another

2.4.2. I/O Protection

- To prevent (نمنع) a user from performing illegal I/O:

* We define all I/O instructions to be privileged instructions.

Thus user cannot issue I/O instructions directly, they must do it through the O.S

يجب ان تكون جميع ايعازات I/O ايعازات مميزة.
وبالتالي لا يمكن للمستخدم إصدار ايعازات I /O بصورة مباشرة وانما يجب عليه القيام بذلك من خلال O.S

- For I/O protection to be complete:

لكي تكتمل حماية الـ I/O:

* We must be sure that a user program can never gain control of the computer in monitor mode.

يجب أن نكون متأكدين من أن برنامج المستخدم لا يمكن أبدا ان يحصل على سيطرة الحاسبة في الـ monitor mode

2.4.3. Memory Protection

- To ensure correct operation: we must protect the **interrupt service routines** in the O.S from modification.

لضمان التشغيل الصحيح :يجب حماية **interrupt service routines** في O.S. من التعديل.

1- We must protect the **interrupt vector** from modification by a user program.

2- Also we must protect the **interrupt service routines** in the O.S from modification..

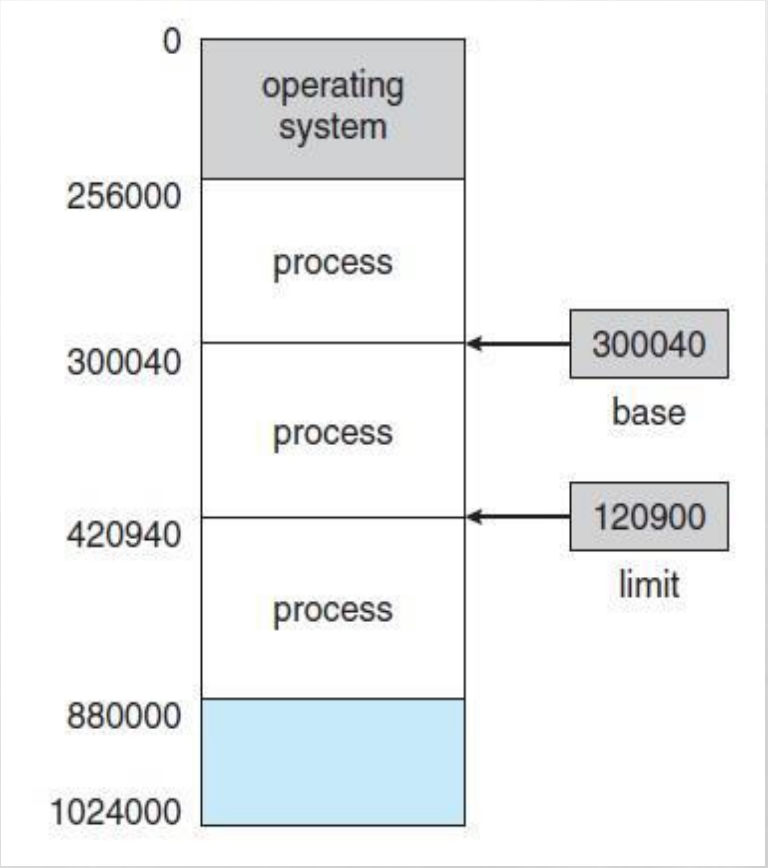
- What we need to separate each program's memory space, the ability to determine the range of legal addresses that the program may access, and to protect the memory outside that space.

نحتاج الى فصل مساحة ذاكرة كل برنامج، والقدرة على تحديد نطاق العناوين التي قد يصل إليها البرنامج، وحماية الذاكرة خارج تلك المساحة.

Memory Protection Cont.

- This protection can provide by using two registers usually a **base** and a **limit**

- The **base** register holds the smallest legal physical memory address
  - The **limi**t register contains the size of the range



- Example:

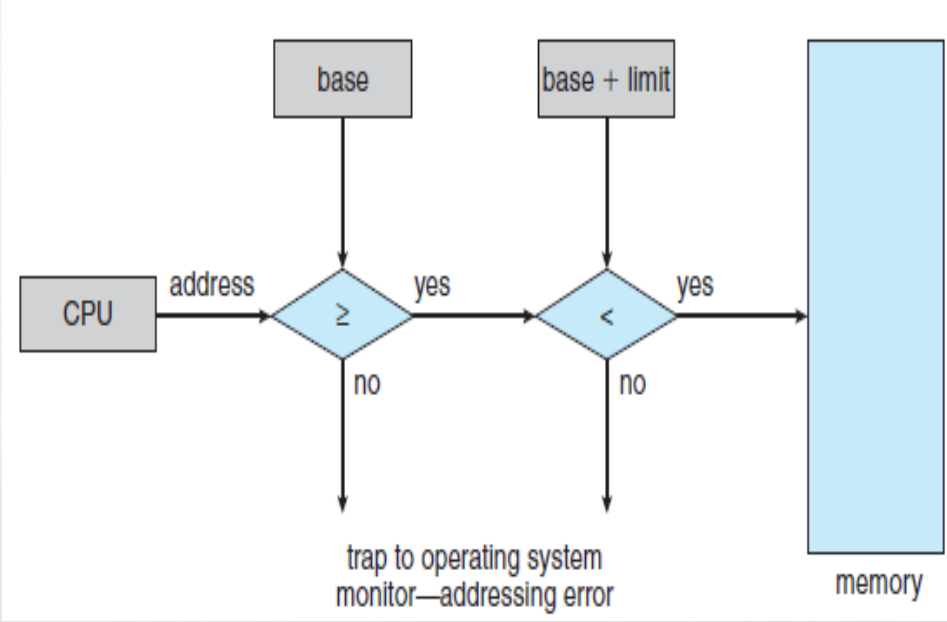
Base register is 300040,

Limit register is 120900

Then the program can legally access all addresses from 300040 through 420940 (base + limit).

Memory Protection Cont.

- The CPU comparing (تقارن) every address generated in user mode with registers accomplishes this protection



- Any attempt by a program executing in user mode to access monitor memory or other user's memory results in a trap to the monitor which treats the attempt as a fatal error

أي محاولة من قبل برنامج تنفيذ في user mode للوصول إلى ذاكرة خاصة بالنظام أو ذاكرة المستخدمين الآخرين ينتج عنه trap إلى جهاز العرض الذي يعامل على انه خطأ فادح

- This scheme prevents the user program from modifying the code or data structures of either the O.S or other users.

يمنع هذا النظام برنامج المستخدم من تعديل ال code او data structures لكل من O.S أو المستخدمين الآخرين.

2.4.4. CPU Protection

- The third piece of the protection is ensuring that the O.S maintains control

الجزء الثالث من الحماية هو ضمان أن يحافظ الـ O.S على السيطرة

- We must prevent a user program from an infinite loop, and never returning control to the O.S

يجب منع برنامج المستخدم من الدخول في infinite loop و لا يرجع السيطرة الى O.S

- To achieve this goal we can use **a timer,** a timer can be set to interrupt the computer after a specified period. The period may be fixed (1/60 second) or variable (from 1 millisecond to 1 second).

يمكن تعيين timer لمقاطعة الحاسبة بعد فترة محددة .قد تكون الفترة ثابتة 1/60 ثانية أو متغيرة من 1مللي ثانية إلى ثانية واحدة

- To control the **timer:** The O.S sets the counter, according to fixed-rate clock. Every time that the clock ticks the counter is decremented. When the counter reaches (0) an interrupt occurs, and control transfers automatically to the O.S, which may treat the interrupt as a fatal error or may give the program more time.

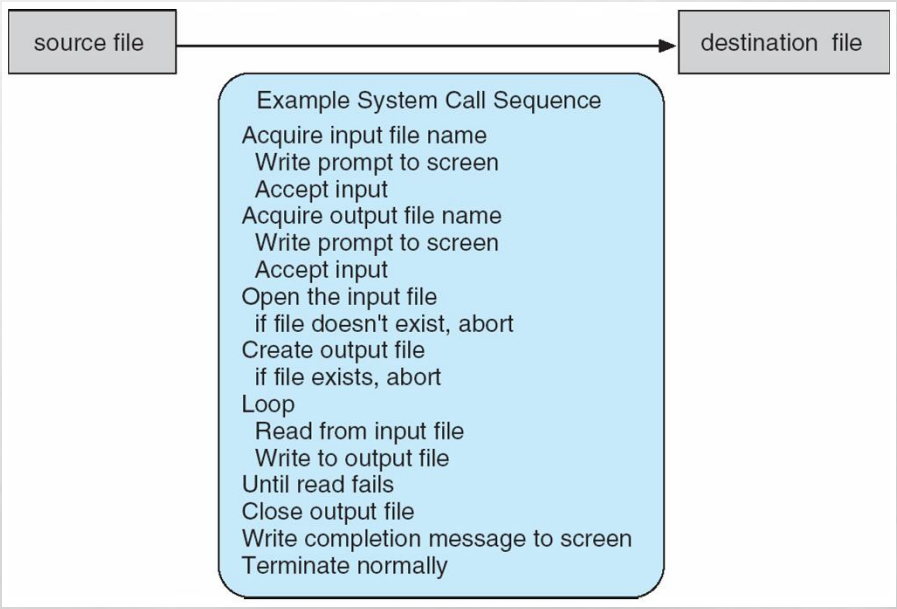
الـO.S يضع قيمة محدد في counter وفقا لمعدل clock ثابت .في كل مرة يحدث ال clock يتم تخفيض ال counter وعندما يصل الى (0) يحدث الـ interupt ، ويتم نقل التحكم تلقائياً إلى O.S، والذي قد يعامل المقاطعة كخطأ فادح أو قد يعطي البرنامج المزيد من الوقت.

2.5. System Calls

- **System calls** provide an interface to the services made available by an operating system

  These calls are generally available as:

  - Routines written in C and C++,
  - Although certain low-level tasks (for example, tasks where hardware must be accessed directly) may have to be written using assembly-language instructions



- From the example you can see
  - Even simple programs may make heavy use of the operating system.
  - Systems execute thousands of system calls per second.

End of Chapter 2