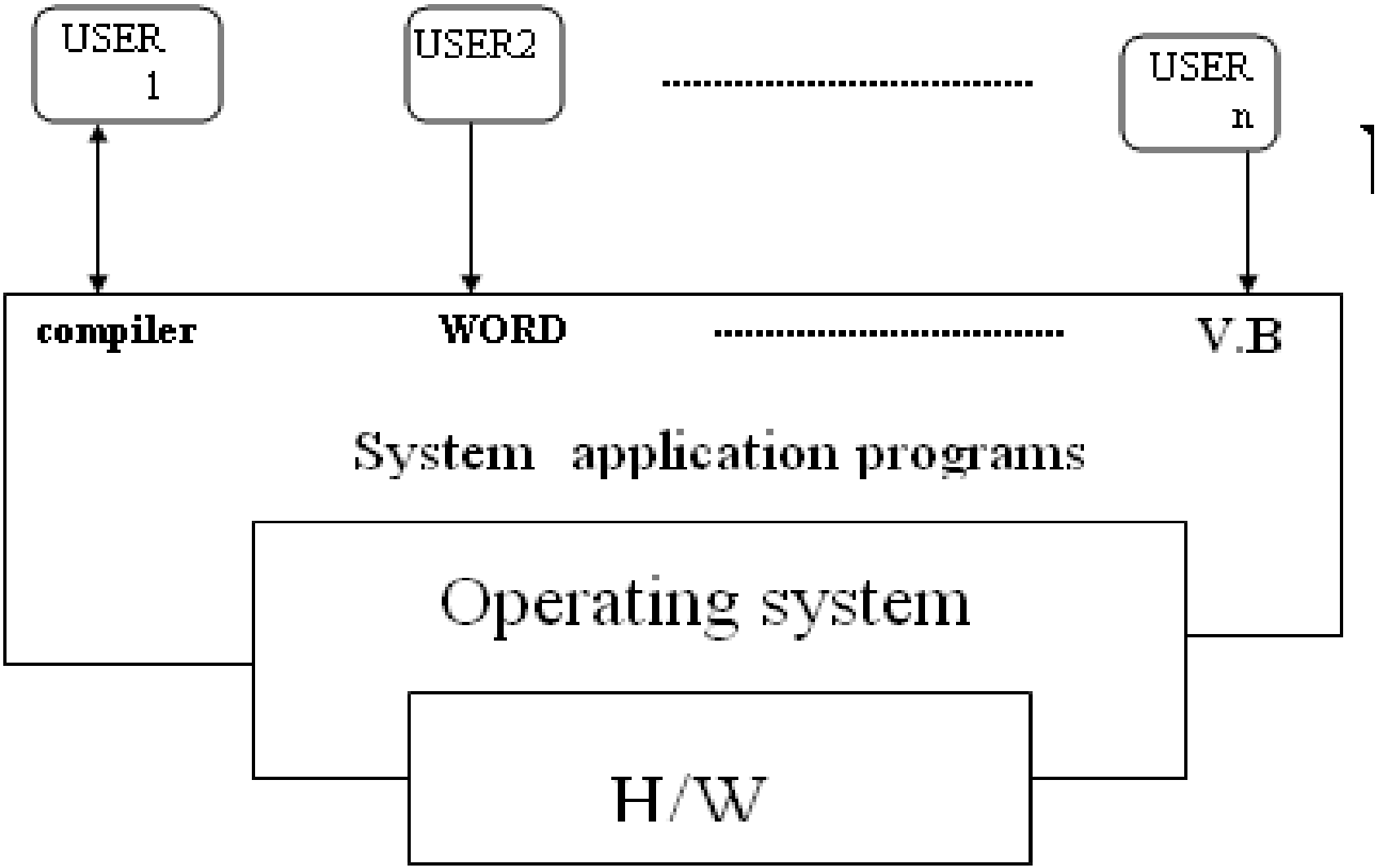
حقيبة تعليمية عن نظام التشغيل O.S

اعداد
حاضر عبد ابراهيم

تعريف نظام التشغيل Operating system

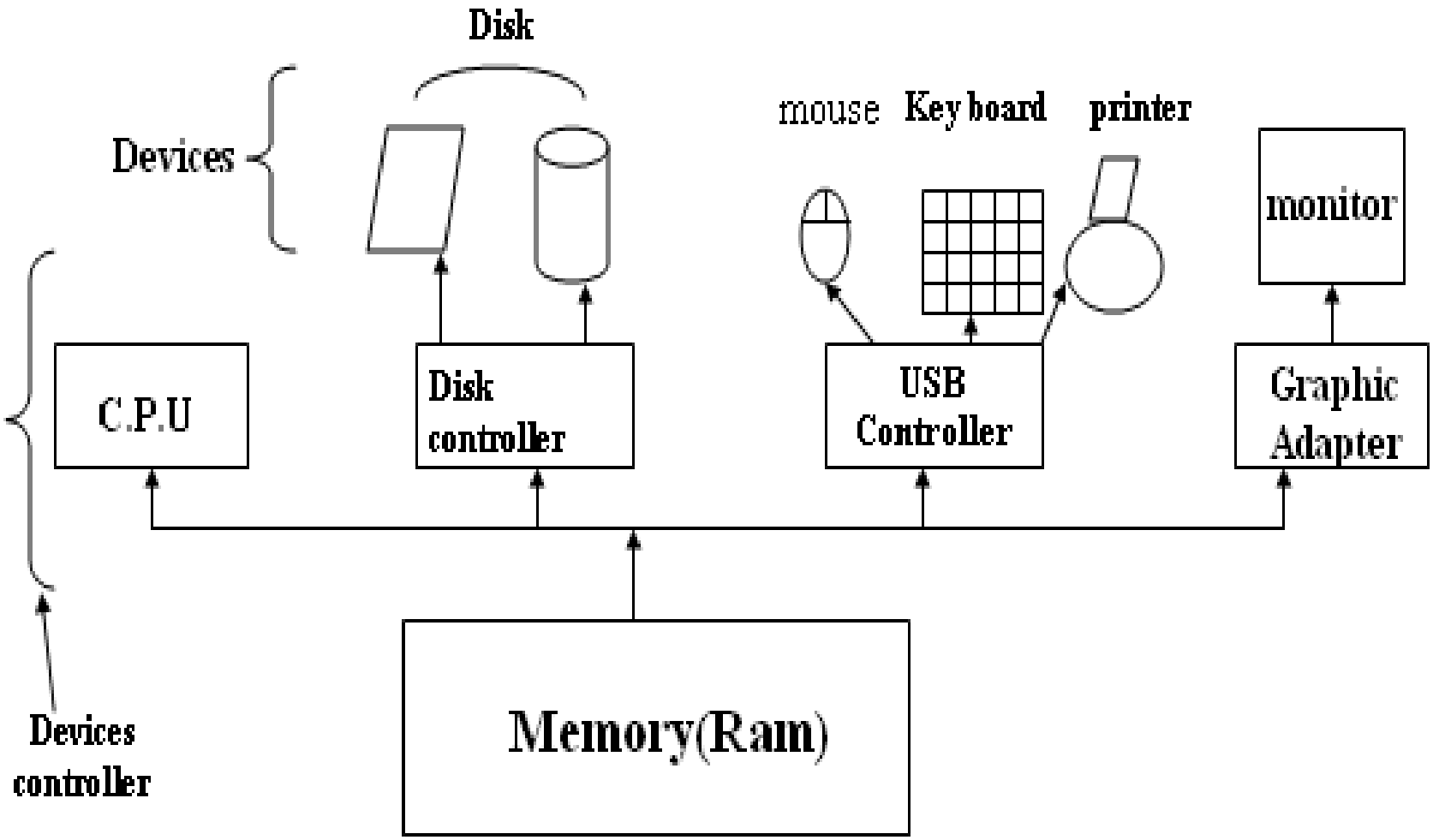
هو برنامج يعمل على ادارة مكونات الحاسبة المادية(H/W) وهو كذلك يعمل كأساس او قاعدة لتنفيذ البرامج التطبيقية ويكون بذلك الوسيط بين مستخدم الحاسبة و اجزاء المادية وكما في الشكل التالي

:(COMPUTER COMPONENTS)


USER 1
USER2
USER n
compiler
WORD
V.B
System application programs
Operating system
H/W

Computer system operation عملية تشغيل نظام الحاسبة

يضم نظام الحاسبة ذات الإغراض العامة (الحاسبة الغير محددة بعمل معين كالحاسبات لأغراض السيطرة)
(General.purpose.computer.system) وكذلك عدد من كارتان السيطرة وعلى الأجهزة (Device controller) والمرتبطة خلال الموصل العام والذي يوفر عملية الدخول للمشاركة في الذاكرة كما في الشكل التالي:


Disk
Devices
mouse
Key board
printer
monitor
C.P.U
Disk controller
USB Controller
Graphic Adapter
Devices controller
Memory(Ram)

في أنظمة الحاسبات الحديثة كل كارت سيطرة(Devices controllers) مرتبط مع نوع محدد من الأجهزة(أجهزة السمع،سائق الأقراص،شاشة العرض) تستطيع كارتات السيطرة على الأجهزة التنفيذ وبشكل متزامن والتنافس على الذاكرة. ولتأمين الدخول وبشكل متوالي إلى الذاكرة (shared memory) المشتركة يعتمد على(memory controller) كارتات السيطرة على الذاكرة لتأمين ذلك من خلال الدخول المتزامن إلى الذاكرة .

كيفية بدء تشغيل الحاسبة

How a computer to start running

عندما يتم تشغيل الكهرباء وتبدأ بتشغيل الحاسبة يكون برنامج التشغيل مخزون في ذاكرة القراءة(Rom) ويسمى هذا البرنامج ببرنامج التشغيل الأولي(Initial program) أو ما يسمى بـ (Boot strap program) ويطلق عليه مصطلح (Term firm ware)

عملية التشغيل تعمل على تصفير كل مكونات الحاسبة من الذاكرة إلى كارتات السيطرة وكذلك الـC.P.U أي يتم حذف البيانات الموجودة في ذاكرة الـRam وعليه يجب على برنامج الـ (Boot strap) أن يعلم كيفية تحميل نظام التشغيل (O.S Kernel) إلى الذاكرة الرئيسية وتشغيل الحاسبة بشكل فعال ويتم ذلك من خلال :

*يقوم برنامج الـ(Boot strap)بتحديد موقع نظام التشغيل والذي يكون عادة على (Hard Disk)وبعدها يحدد ما يسمى ب لب (Kernel)نظام التشغيل الذي يقوم بتحميله إلى ذاكرة (Ram) بعد ذلك يقوم نظام التشغيل بتنفيذ أو عملية التي تسمى"Lint" وبعدها ينتظر الأوامر.

وبذلك يكون جهاز الكومبيوتر فعال من خلال إتمام عملية تشغيل نظام أل(O.S)

نظام معالجة الدفعات

يعد البرنامج التعليمي (program) الشئ المهم من خلال تنفيذ تعليماته(Instruction) وبدون ذلك يصبح لا فائدة منه ويتم تنفيذ هذه التعليمات من خلال الـ(c.p.u) يسمى البرنامج في حالة التنفيذ بالعملية(process) ومن أمثلة العملية(process) هي المفهرس(compiler) والمفسر (Interpreter) كذلك معالج النصوص (Word processing) أثناء تنفذه من قبل أي شخص يدعى بالعملية (process).كما إن عملية إرسال تقرير الى الطابعة يمكن إن يسمى بالعملية.

وعليه يمكن أن نفهم العملية(process) على أنها job (code & data) أو برنامج المشاركة الزمنية(time-shared program).

تحتاج العملية(process) مصادر مؤكدة مثل c.p.u time، الذاكرة memory، الملفات files، وأجهزة الإدخال والإخراج I/O devices.

لانجاز مهمتها تعطي هذه المصادر الى العملية (process) عند تكوينها أو تحدد لها أثناء تنفيذها كذلك تمرر لها البيانات data من خلال عملية الإدخال (Input) فمثلا عملية عرض حالة الملف (file status) على الشاشة يتم ذلك من خلال إدخال اسم الملف وبعد ذلك سوف يتم تنفيذ تعليمات معينة وبعدها سوف يتم عرض حالة الملف وعند انتهاء العملية سوف يقوم (O.S) بإعادة استدعاء أي مصدر ومما سبق فان البرنامج الذي يخزن على الـ Hard أو على الـFlash أو في الـCD هو ليس عملية (process) لأنه ليس في حالة تنفيذ(Running) أي ان محتوى الملف المخزون على القرص يمثل برنامج يسمى بـpassive entity وحدة معطلة.

لكن البرامج التي تحت التنفيذ (Run) تسمى بالعملية (process) وتسمى بمصطلح Active entity وحدة أو كيان فعال. وعليه تمتلك العملية process ذات المسار المنفرد(Single-threaded)عداد برنامج (Program counter) والذي يحدد التعليمة القادمة المقدمة للتنفيذ ويكون تنفيذ تعليمات العلمية بشكل تسلسلي (sequentially). أي ان الـc.p.u ينفذها تعليمة بعد تعليمة يمكن ان يظم البرنامج أكثر من عملية(process) ويكون بذلك تنفيذ كل عملية على حدى تدعى العملية "هي وحدة العملية في النظام ألحاسباتي "مثلا يظم النظام مجموعة من العمليات(processes) بعض من هذه العمليات تخص نظام التشغيل والباقي تخص المستفيدين(Users).

كل هذه العمليات يمكن تنفيذها على وجه التحديد بشكل متزامن من خلال تناوب الـ C.P.U المنفرد في تنفيذها. وعليه يكون نظام التشغيل مسؤول عن الفعاليات التالية والمرتبطة بإدارة العمليات :

- تكوين وحذف عمليات المستفيد والنظام.
- تعليق أو إيقاف تنفيذ العملية أو إعادة تنفيذها.
- توفير البدء لتنفيذ العمليات المتزامنة .
- توفير آليات لعمليات الاتصالات .
- توفير آليات لحل مشكلة الاختناق (Dead lock).

وحدة العملية(Instruction unit)

أدارة الذاكرة الرئيسية Main memory management
(Ram management)

تعتبر الذاكرة الرئيسية او ما يسمى بالذاكرة العشوائية (Ram) هي مركز العمليات في أنظمة الحاسبات الحديثة.تعتبر الذاكرة منظومة (مصفوفة) ذات بعد واحد كبيرة او واسع من البايتات او الكلمات.وكل كلمة او بايت يكتلك عنوان خاص به وعليه تعتبر الذاكرة الرئيسية:عبارة عن مستودع للبيانات المشاركة ذات الدخول السريع من قبل الـc.p.u وأجهزة الإدخال والإخراج(I/O devices).

حيث ان الـC.P.U تعتبر المصدر الذي يتعامل بشكل مباشر مع الذاكرة الرئيسية من خلال عملية الجلب او ما يسمى بالـ Instruction-fetch cycle أي دورة جلب التعليمة

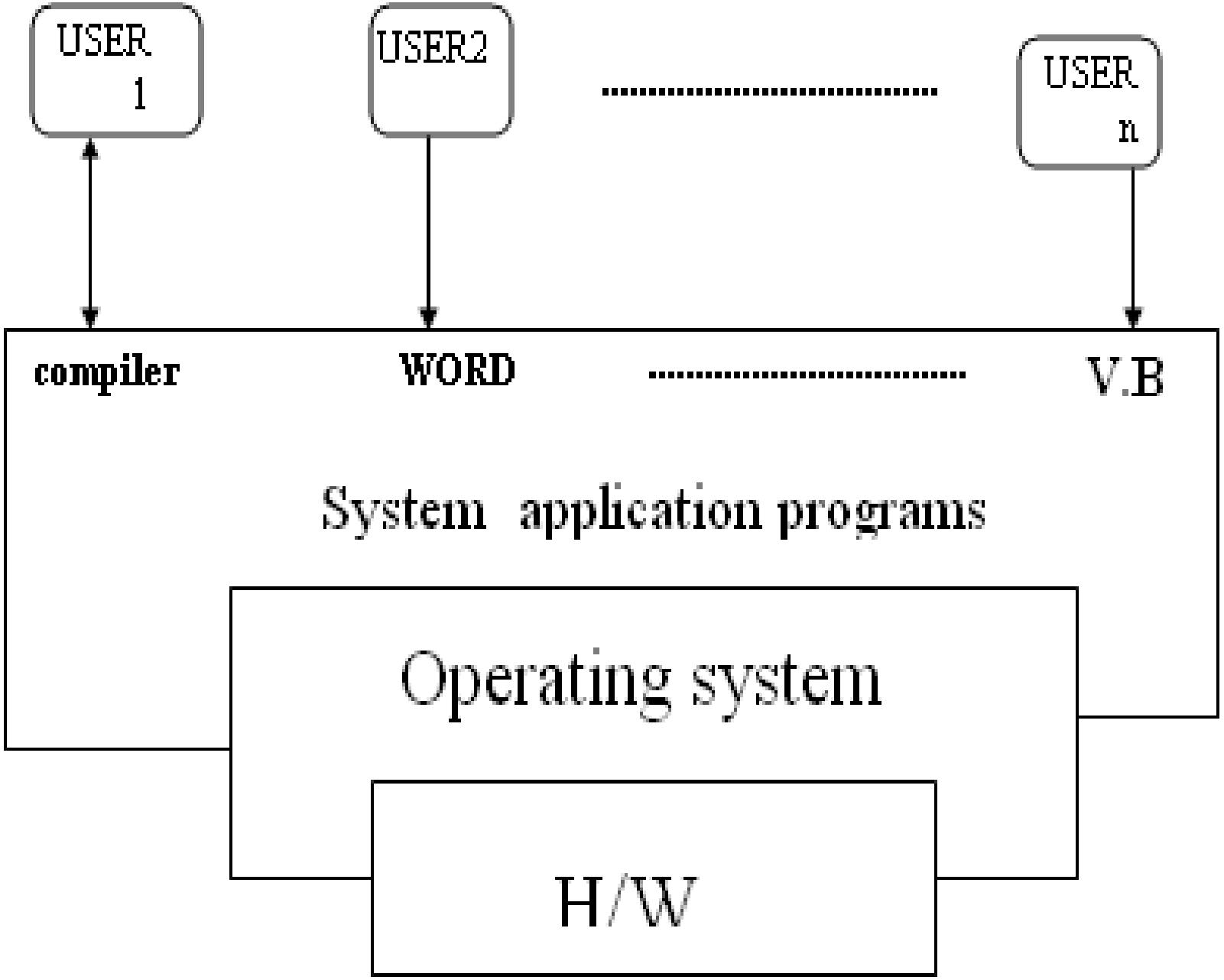
وبشكل عام تعتبر الذاكرة الرئيسية هي المخزن الكبير الذي يستطيع الـ C.P.U الدخول له مباشرة مثلا:عندما يعالج الـC.P.U البيانات المخزونة على الـ H/Dيجب نقل هذه البيانات الى الذاكرة الرئيسية وبعد ذلك يستطيع معالجتها ولتنفيذ البرنامج يجب أن يطبع البرنامج(Mapped) من خلال عناوين مطلقة ثم بعد ذلك يحمل الى الذاكرة الرئيسية (Loaded to Memory) .

وعند الانتهاء من تنفيذ البرنامج المساحة المشغولة من الذاكرة سوف تعلن كمساحة فارغة يستطيع البرنامج اللاحق استغلالها وللتطوير وتحسين استخدام الـC.P.U وسرعة الحاسبة يستفيد منها المستخدم عملية أنظمة الحاسبات ذات الإغراض العامة على حفظ عدد من البرامج في الذاكرة.وكما اسلفنا بما يسمى بـmulti programming. في الذاكرة وبهذا أصبحت الحاجة ملحة لإدارة هذه الذاكرة بنظر الاعتبار وعليه فان مسؤولية نظام التشغيل اختصت بالفعاليات التالية والمتعلقة بإدارة الذاكرة الرئيسية وهي:-

- حف المسارات للأجزاء المستخدمة من الذاكرة وتحديد المستخدمين لهذه الأجزاء.
- تحديد العمليات والبيانات الوافدة الى الذاكرة والخارجة منها.
- تحديد المساحات المطلوبة وإلغاء المساحات المحددة والتي افرغ من استخدامها.

المؤقت Timer

يجب ان نؤمن ان نظام التشغيل يمتلك السيطرة على الـC.P.U
ويجب ان نمنع برامج المستفيد (User programming) من ان تبقى فترة طويلة وغير محددة في التنفيذ وكذلك نمنع حالة عدم استدعاء (calling system services) وكذلك منع حالة عدم إرجاع السيطرة الى(O.S) ولانجاز كل ما ذكر سابقاً نحن نستطيع استخدام الـ"Timer" يعمل الـTimer على جعل القطع يحدث بعد فترة محددة وهذه الفترة ممكن ان تكون ثابتة مثلا(١/٦٠)ثانية مثلا (١ ملي سكند الى ١ سكند).(1 millisecond to 1 second)
وعليه نستخدم الـTimer لمنع حالة حدوث تنفيذ برنامج من قبل المستفيد ولفترة طويلة وغير محددة.


USER 1
USER2
USER n
compiler
WORD
V.B
System application programs
Operating system
H/W

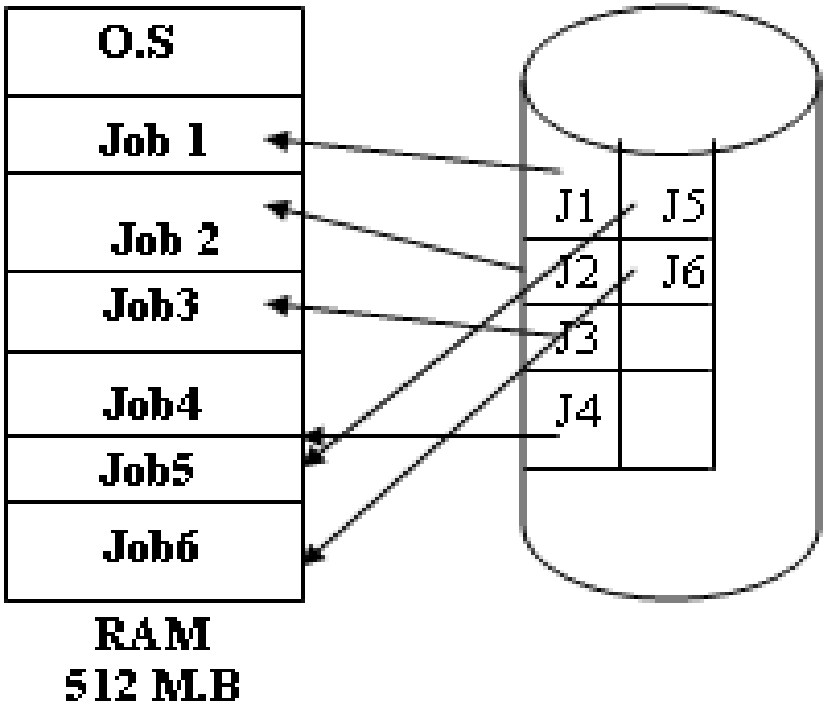
نظام البرامج المتعددة Multiprogramming system

يعتبر نظام البرمجة المتعددة من أهم أوجه نظام التشغيل الحديث (Modern Operating system) بحيث يعطي القدرة على تنفيذ عدة برامج في آن واحد وهذا ما يميزه عن أنظمة التشغيل القديمة (single user O.S) بشكل عام تعطي هذه القدرة على جعل C.P.U أو أجهزة الإدخال والإخراج مشغولة على طول الوقت "أي لا يوجد هناك وقت فراغ لـC.P.Uأو I/O Devices" البرمجة المتعددة تعمل على زيادة استخدام الـC.P.U من خلال تنظيم الـJob (Code & Data) ولذلك على طول الوقت هناك Job ينتظر التنفيذ من قبل الـC.P.U.

يمكن توضيح فكرة البرمجة المتعددة على النحو التالي:

يقوم الـO.S بحفظ عدد من الـJobs في الذاكرة الرئيسية (Ram) وهذه الـJobs هي جزء من الـJobs التي تم حفظها بمكان محدد من القرص الصلب (H/D) والذي يسمى Job pooling وكما موضح بالشكل التالي:

JOB SCHEDULING



المناقلة من/إلى SWAPPING H/D IN/OUT

Memory layout for multiprogramming system

حيث يقوم ما يسمى Job scheduling باختيار الـJOB الجاهز للتنفيذ الى الذاكرة (RAM) وحسب حجم الذاكرة المسموح به.

Job pooling: هو مكان محدد من القرص الصلب يأخذ شكل الذاكرة الرئيسية ويسمى بالذاكرة الثانوية (secondary memory) حيث يتم حفظ جميع الـ Jobs في هذا المكان وتكون سرعته أبطئ من الذاكرة الرئيسية .لأنه يعتبر ذاكرة خارجية الربط عن طريق الـData Cable .

يقوم نظام التشغيل باختيار عدد من الـJobs الموجودة في الذاكرة لتقديمها الى الـC.P.U للتنفيذ والذي يقوم بعملية اختيار أو تحديد الـJobs الذي سوف تنفذ من قبل الـC.P.U وهو الـC.P.U Scheduling يحدث انتظار للـJob المنفذة على الـC.P.U بسبب حاجتها الى بعض المكتبات التي تم استخدامها من قبل الـJobs أخرى مثل I/O operation.

*المقصود بالمكتبات: هي مكتبة I/O، Math والتي تستخدم إيعاز (#Include <أسم المكتبة>) المستخدمة بلغة C++.

وبذلك لا يوجد وقت فراغ للـ C.P.U أما في أنظمة التشغيل القديمة (Single user O.S) مثل MS-DOS فيبقى الـC.P.U بدون عمل (Idle).

طريقة عمل نظام التشغيل في البرمجة المتعددة:

يقوم الـO.Sببساطة بالتحول بين البرامج
المنفذة"Switches"الـJobsوالهيئة للتنفيذ من قبل الـC.P.U
أي ينتقل الـO.S بين هذه الـJobs لكي ينفذها من قبل
الـC.P.U وبسرعة عالية جدا بحيث يعطي انطباع للمستفيد
بان هذه الحاسبة أو الـC.P.U مخصصة له فقط.

Time Sharing system نظام المشاركة الزمنية

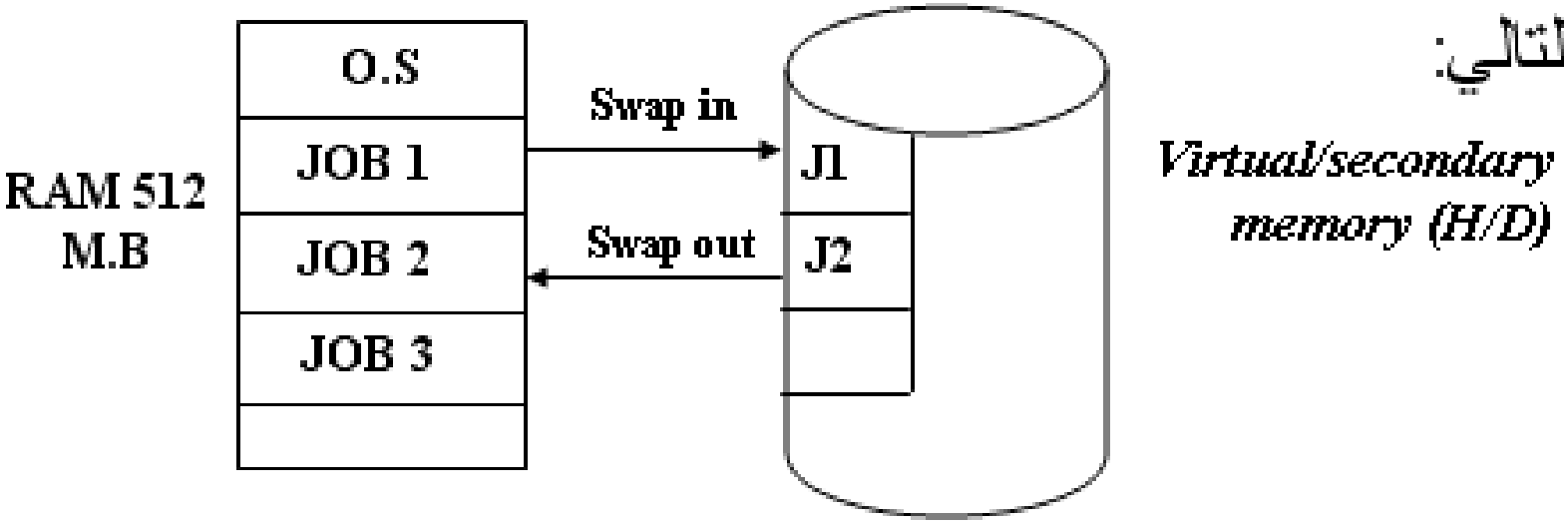
يسمى نظام المشاركة الزمنية بنظام المهام المتعددة (Multi Tasking system) يزود نظام البرمجة المتعدد البيئة الملائمة لأستخدم مصادر النظام المختلفة (Various system resources) مثلا : (Memory, I/O devices, C.P.U) وبشكل كفوء ولكن يوفر للمستفيد التفاعل مع نظام الحاسبة .يعتبر نظام المشاركة الزمنية"نظام تعدد المهام " (Multi Tasking system) :امتداد منطقي لنظام البرمجة المتعددة حيث يتم في نظام المشاركة الزمنية

(multi tasking system)تنفيذ الـJobs من قبل الـC.P.U من خلال التحول عبر كل هذه الـJobs و لكن هذا التحول (Switches) يحدث بشكل سريع ومتكرر بحيث يشعر المستفيد (User) يستطيع التفاعل مع أي برنامج أثناء تنفيذه.

متطلبات نظام المشاركة الزمنية

نظام حاسبة متفاعل بحيث يعطي المستفيد التعليمات للحاسبة من خلال أجهزة الإدخال (Key board ,Mouse,.....) ويقوم نظام التشغيل بتنفيذها وعليه يكون هناك استجابة زمنية (Response Time) قصيرة جدا تصل الى اقل من الثانية .
يسمح نظام التشغيل ذا المشاركة الزمنية (Time shared O.S) لعدد من المستفيدين ليتشاركوا في الحاسبة بشكل متزامن .وعليه كل أمر بنظام المشاركة الزمنية يميل لان يكون قصيرا أي يحتاج الـC.P.U الى زمن قصير لكل مستفيد.
وعليه نظام التشغيل ذا المشاركة الزمنية يستخدم جدولة الـ C.P.U (C.P.U scheduling) والبرمجة المتعدد لتزويد أي مستفيد بحصة قصيرة أو صغيرة من الحاسبة ذات المشاركة الزمنية(Time_Shared_Computer).

وعليه يجب أن يوفر نظام التشغيل أو يؤمن زمن استجابة (Response Time) معقول لنظام المشاركة الزمنية (Time Sharing System) ويمكن تأمين هذه من خلال عمله،يسمى (Swapping In/Out) "مناقلة"بين الذاكرة الرئيسية والذاكرة الثانوية (Secondary Memory) أو تسمى بالذاكرة الافتراضية (Virtual Memory) حيث يتم نقل العملية (Process) التي لم يكتمل تنفيذها من الذاكرة الرئيسية الى الذاكرة الافتراضية وبالعكس وحسب الشكل التالي:



Multiprocessing System نظام المعالجة المتعددة

يمكن قد تبين نظام الحاسبة بعدت طرق مختلفة ويكون هذا الترتيب أو التصنيف بشكل عام معتمدا على عدة معالجات المستخدمة في كل حاسبة وعليه يمكن تصنيفه إلى ثلاثة أنواع :

1-Single-processor system:

يشمل هذا النوع الحاسبات الشخصية P4 والتي تظم (P4) c.p.u واحد فقط.

2-Multiprocessor System:

ويسمى هذا النوع بـparallel system أو Tightly coupled system والذي يعتبر متزايد في أهميته لكون أن أي حاسبة تظم أكثر من C.P.U ويقسم إلى نوعين:

١ -متعدد المعالجات الغير متناظر

Asymmetric multiprocessing

بحيث يحدد لكل معالج عمل معين وتحكم العلاقة بين المعالجات لهذا النوع علاقة "السيد بالعبد"

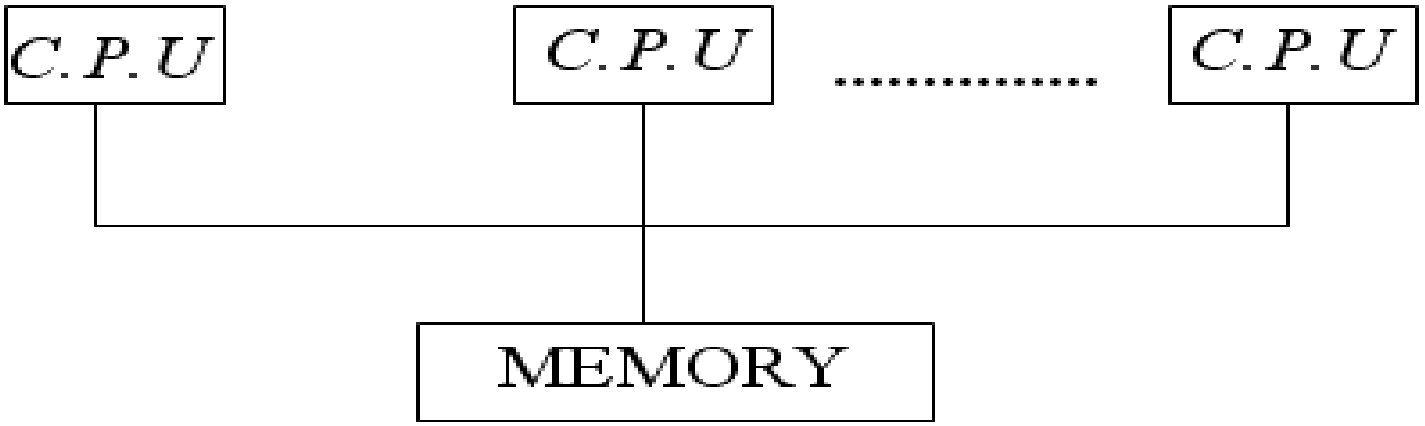
(Master-slave Relationship) بحيث يحدد المعالج السيد (Master) والمسيطر على الحاسبة العمل لبقية المعالجات والتي تسمى بـ Slaves أو تعمل بالأعمال المحددة مسبقا.

٢ -معالجات المتناظرة

Symmetric multiprocessing

وتختلف هذه في طريقة التعامل بحيث كل Processor (C.P.U)تنفذ كل الإعمال من خلال نظام التشغيل .وعليه تكون كل المعالجات هي ند أو Peers ولا تحكمها علاقة السيد بالعبد بحيث تشترك جميع المعالجات (C.P.U) بذاكرة واحدة.كما بالشكل التالي:

S.M.P Architecture



وهي أمثلة الأنظمة التشغيلية التي تتعامل مع هذا النظام هي
Solaris system, commercial version of UNIX system
الفائدة من هذه التقنية (S.M.P) هي تنفيذ عدد من العمليات بشكل متزامن ولا يوجد توقف لتنفيذ أي عملية Process.

N processes ⟶ *N C.P.Us*

٣ ـ النظام العنقودي Clustered system:

يستخدم هذا النظام عدة معالجات لانجاز عمل معين لكنه يختلف عن الأنواع السابقة الذكر يكون مركب من عدة حاسبات مستقلة مرتبطة مع بعض سوية.ويمكن تعريف على أن مشاركة عدة حاسبات بالخزن

(Computer share system) تربط بـ LAN (Local area network) والعنقودي ممكن أن يتكون أو يهيكل بطريقة التناظر أو غير المتناظر.

نظام الوقت الحقيقي Real Time System

يسمى هذا النظام بنظام الغرض الخاص

(Special-Purpose system)

ويظهر في الانظمة التي تستخدم التقنيات التطبيقية ذات الطابع المدني و
العسكري وتكون مظمنة ضمن تطبيق صناعي معين للسيطرة عليه كمحرك السيارة والربونات وكذلك الصواريخ وأجهزة تحسس الهزات والزلازل الارضية ويستخدم بشكل اساسي للتعامل عن بعد مع هذه
الاجهزة والسيطرة عليها بشكل دقيق جدا

هيكلية المخزن Storage Structure

عند تنفيذ البرامج على الحاسبة يجب أولا:حسب أسلوب البرمجة المتعددة إن ينتقل البرنامج إلى الذاكرة الرئيسية (Ram) أما يسمى بالذاكرة العشوائية لكي يكون مستعد لعملية التنفيذ من قبل المعالج (C.P.U) وحسب ما أخذناه سابقا تعد الذاكرة الرئيسية المخزن الكبير بعد القرص الصلب والتي تقدر مساحتها بـأكثر من كيكابايت (1024 M.B) والتي يستطيع المعالج الدقيق (C.P.U) التعامل معها بشكل مباشر وتسمى هذه الذاكرة وبعد تطورها بـD Ram وهي اختصار لـ(Dynamic Random access Memory) والتي تكون على شكل منظومة من الكلمات.

منظومة:هي مصفوفة أحادية البعد تسمى (Array) وكل كلمة لها عنوان خاصة بها.

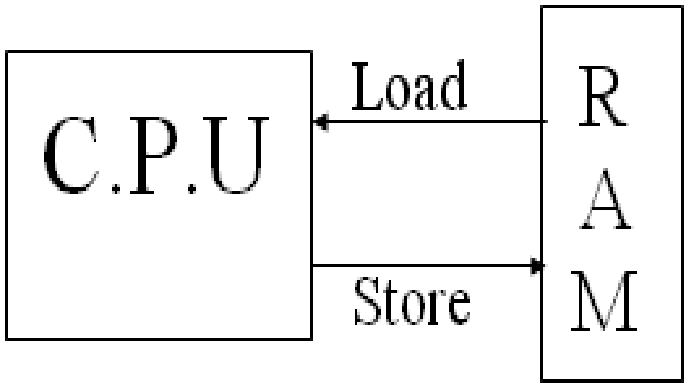
أن عملية التعامل مع الذاكرة يتم من خلال الأمرين (2 function) المتسلسلتين التاليتين Store, Load وتعني تحميل المعلومات او التعليمات (Instructions) في الذاكرة او من الذاكرة حيث:

Load Instruction:تحميل المعلومات من الذاكرة الى المعالج.

Store to address of memory: تعني خزن المعلومات من المعالج للذاكرة.

حيث تعني Load:تحويل الكلمة من الذاكرة الرئيسية الى سجل داخل الـC.P.U بينما .....

Store:تحويل محتويات السجل في C.P.U الى الذاكرة الرئيسية وهذه العملية تسمى بـ(Fetch)جلب وتكون حسب الشكل التالي:



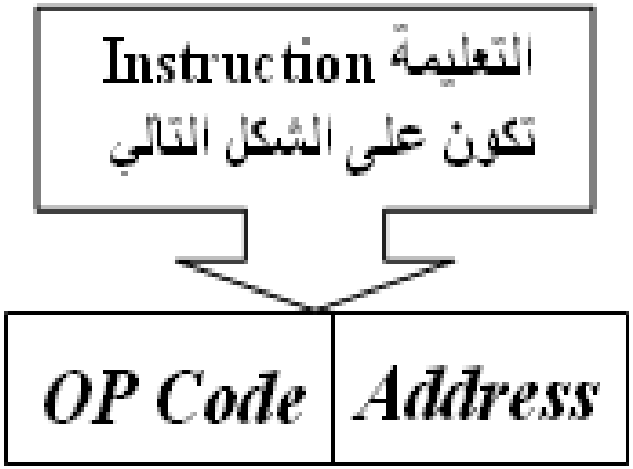
التعليمة او ما يسمى بالـ Instruction :عبارة عن كبسولة مكون من جزأين هما :

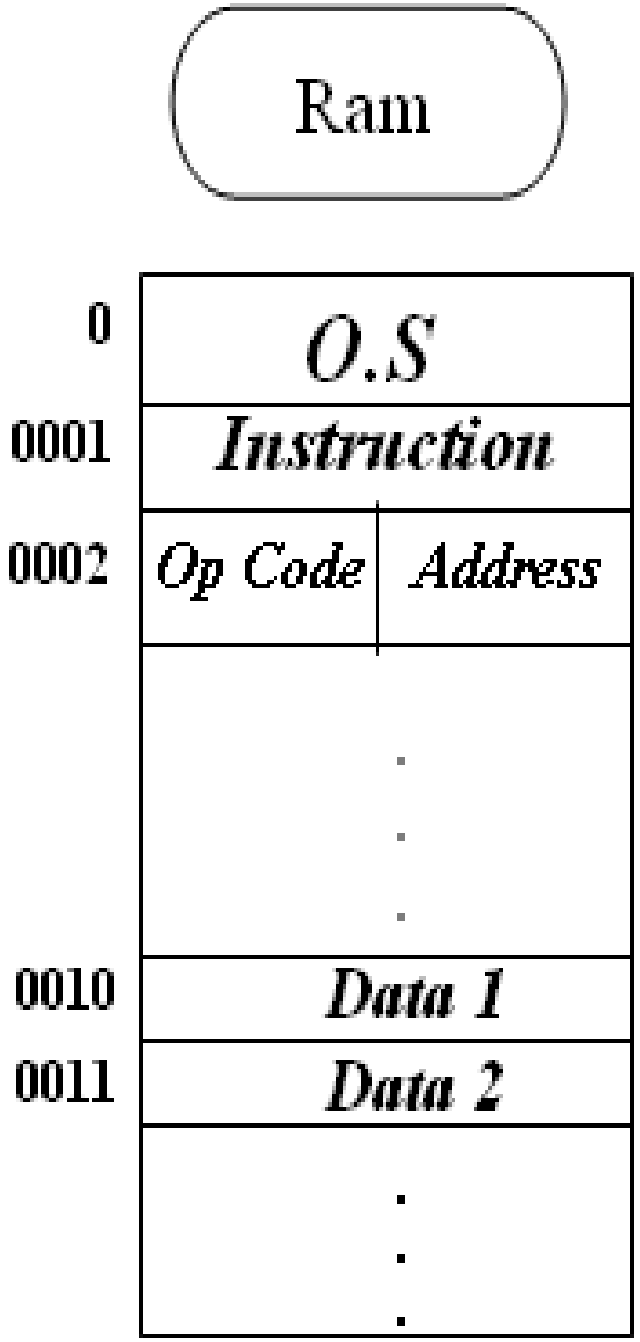
1-OP CODE

تعني تشفير العملية المطلوبة الجزء الأخر.

2-Address:

يعني عنوان الجزء الذي يظم البيانات ويسمى بالـ (Data "Operand") وتقسم الذاكرة الى أجزاء من التعليمات وأجزاء للبيانات وعند تنفيذ البرنامج فان التعليمات تظم تعليمات البرنامج من جزء بيانات البرنامج وكما يأتي بالشكل التالي:





*أما البيانات Data تكون Integer أي أعداد صحيحة وتسمى Operand وحسب معمارية النظام Von Neumann .

تكون دورة تنفيذ التعليمة(Instruction) حسب الخطوات التالية:

١ـتجلب التعليمة من الذاكرة وتخزن في مسجل بـC.P.U.

٢ـالتعليمة يفك الشفرة التابعة لها بـDecode ويمكن أن تكون عملية التشفير هي تحميل بيانات (Operand)من الذاكرة الى المعالج.

٣ـبعد إجراء العمليات في C.P.U يتم خزن النتائج في الذاكرة الرئيسية بعد انتهاء تنفيذ البرنامج من خلال الأمر Store.

وعليه يتكون المعالج C.P.U من الأجزاء التالية:

1-PC : Program Counter مخزن يحمل عنوان التعليمة في الذاكرة.

2-AC : Accumulator مخزن مؤقت لجمع البيانات.

3-IR : Instruction Register مسجل التعليمة

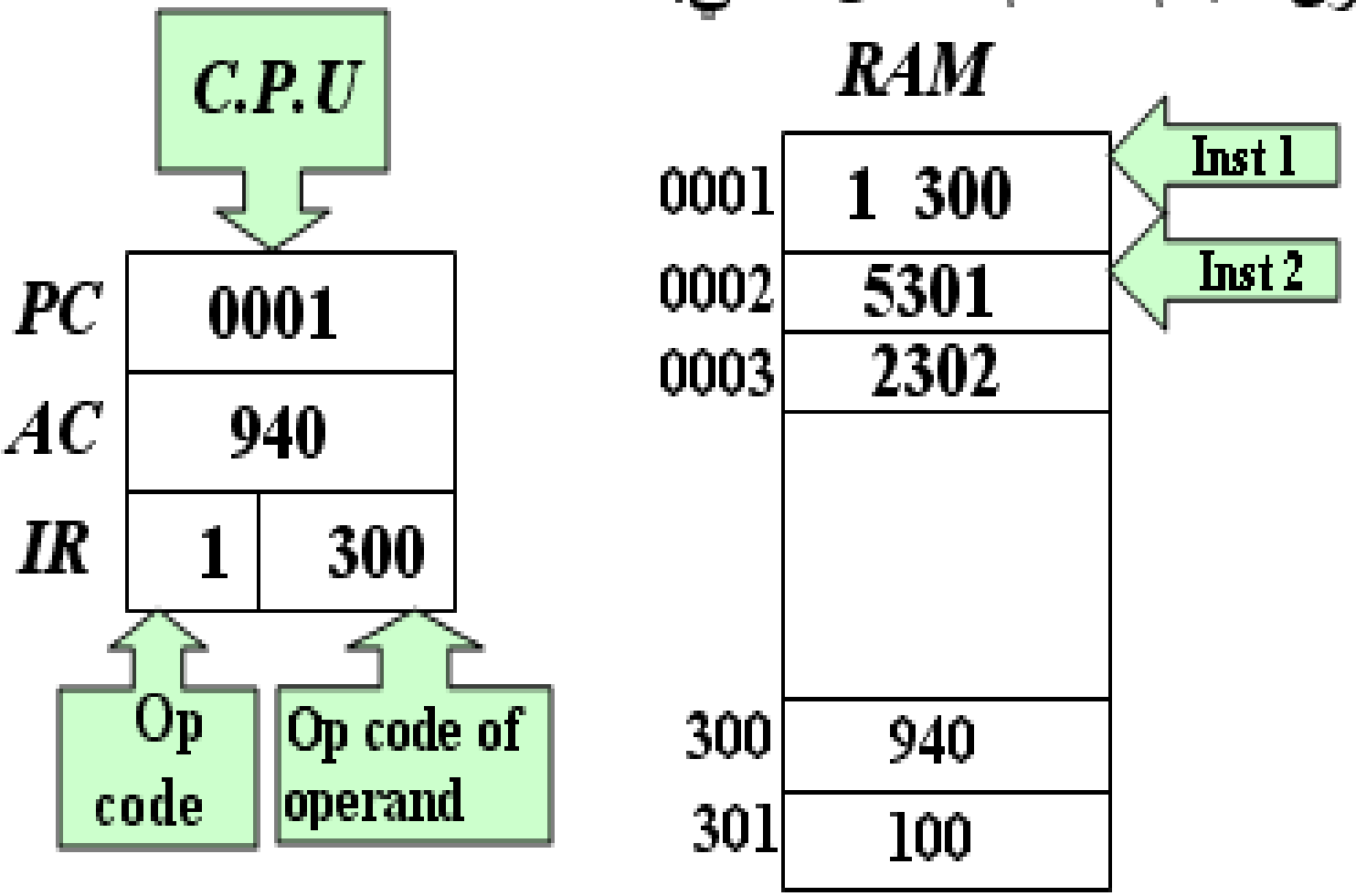
أما العمليات فتكون موجزة حسب الجدول التالي وتكون Hexa Decimal

0001 تعني ***Load*** من الذاكرة إلى PC

0002 تعني **Store** من AC الى Ram

0005 تعني في الذاكرة من آت هو ما مع AC موجود ما جميع

ويكون الجلب حسب الشكل التالي:



تكون عملية الجلب (Fetch) وحسب الشكل السابق:

١-PC=0001 عنوان التعليمة في الذاكرة.

٢-في تحميل التعليمة الى IR ثم فك (Decode) حيث أنها تتكون من

1	300

حيث "1" تعني Load للمعلومة الموجودة بالعنوان "300" في الذاكرة.

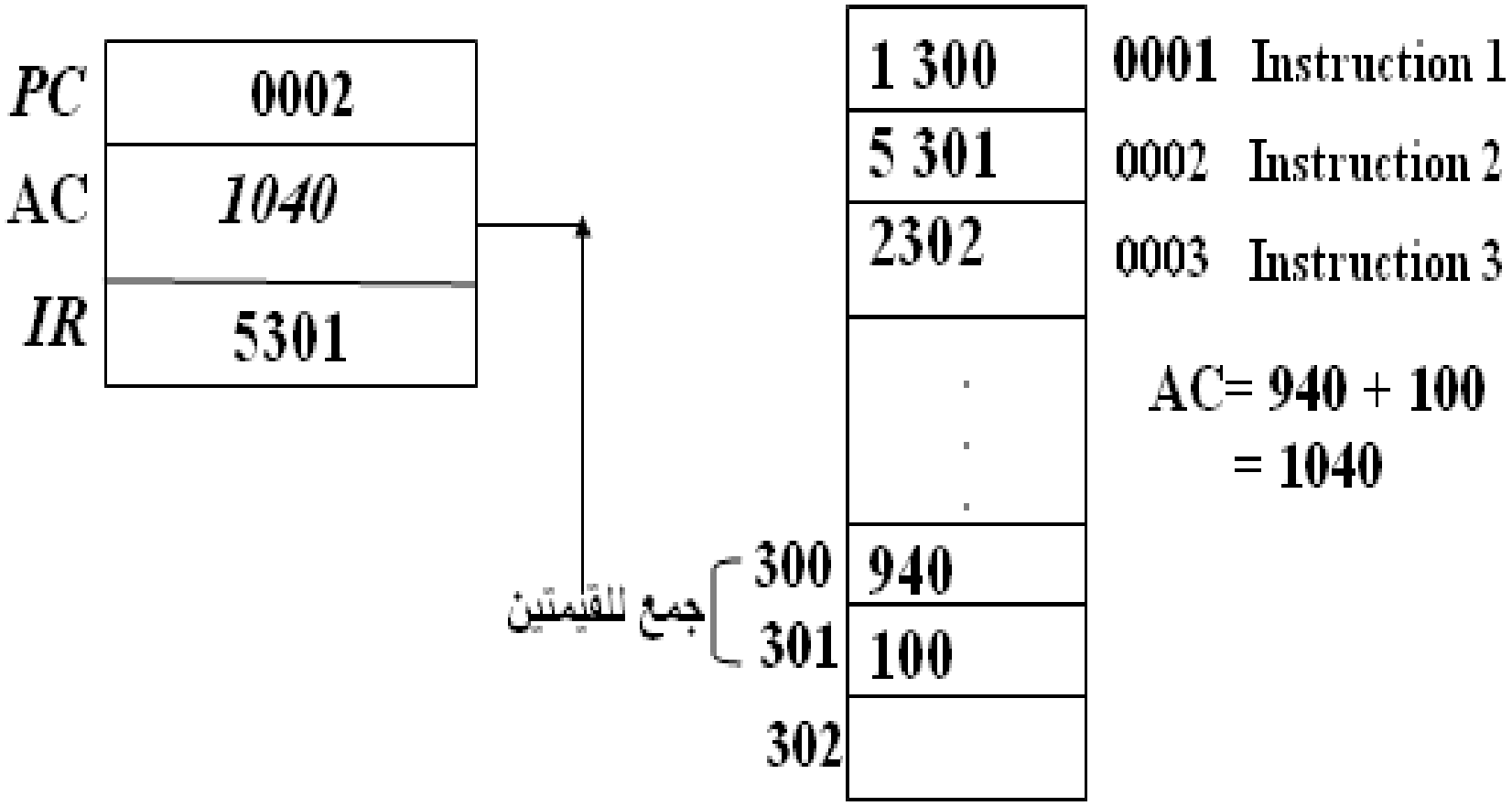
٣-أصبحت هذه التعليمة مخزونة في سجلات الـC.P.U الداخلية بعدها يستمر تنفيذ البرنامج حيث تحوي في الذاكرة أكثر من تعليمة حيث تعاد الخطوات الثلاث السابقة حيث تصبح PC=0002 و

IR=301 5 بعدها تفك الشفرة Decode

OPcode Address

وتعني عملية الجمع القيمة السابقة في AC+ القيمة الحالية

فيكون الشكـــل كالأتي:



أما التعليمة التالية في البرنامج فتكون عملية خزن Store للبيانات الموجودة في AC إلى الذاكرة لأنها تنفذ البرنامج وكما يأتي:

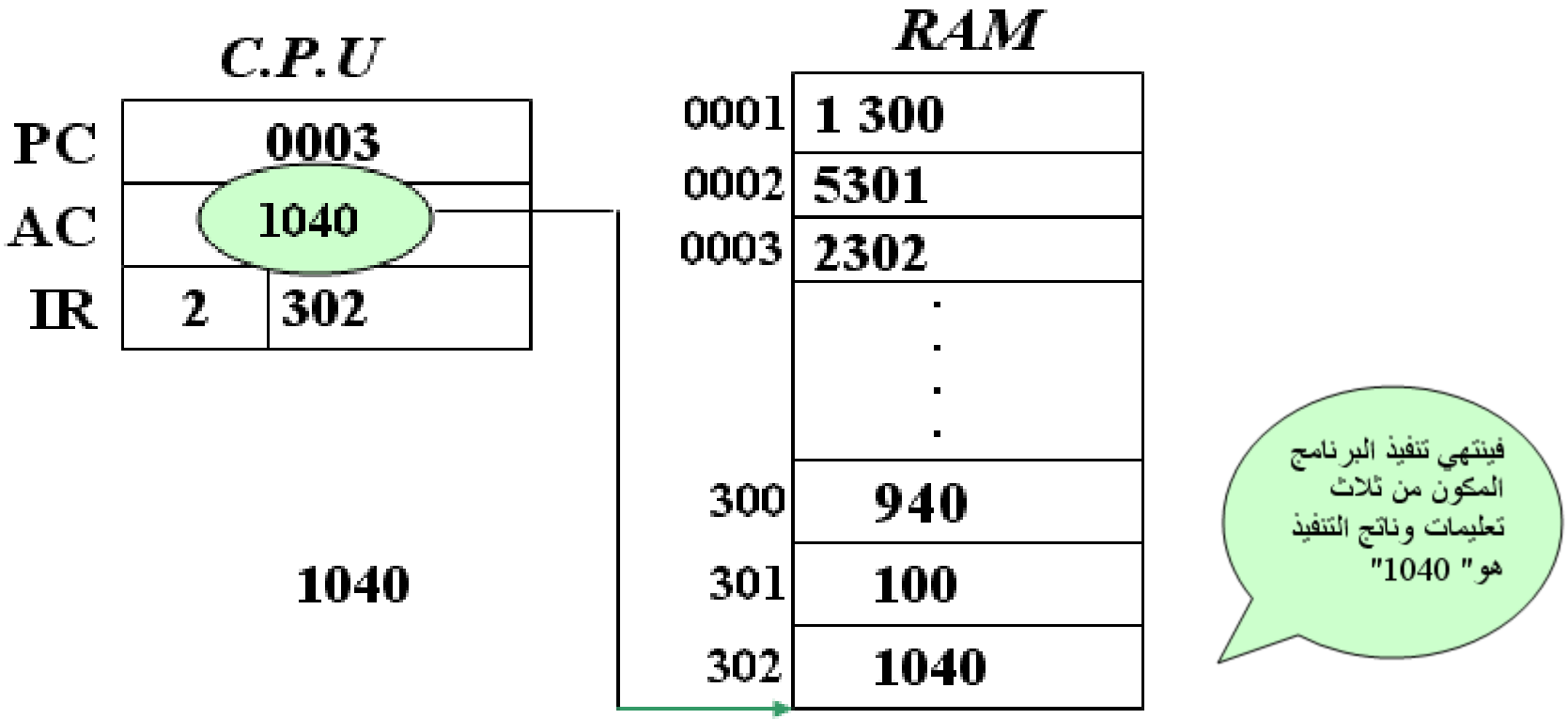
PC = 003

IR = 2302

فان AC=1040

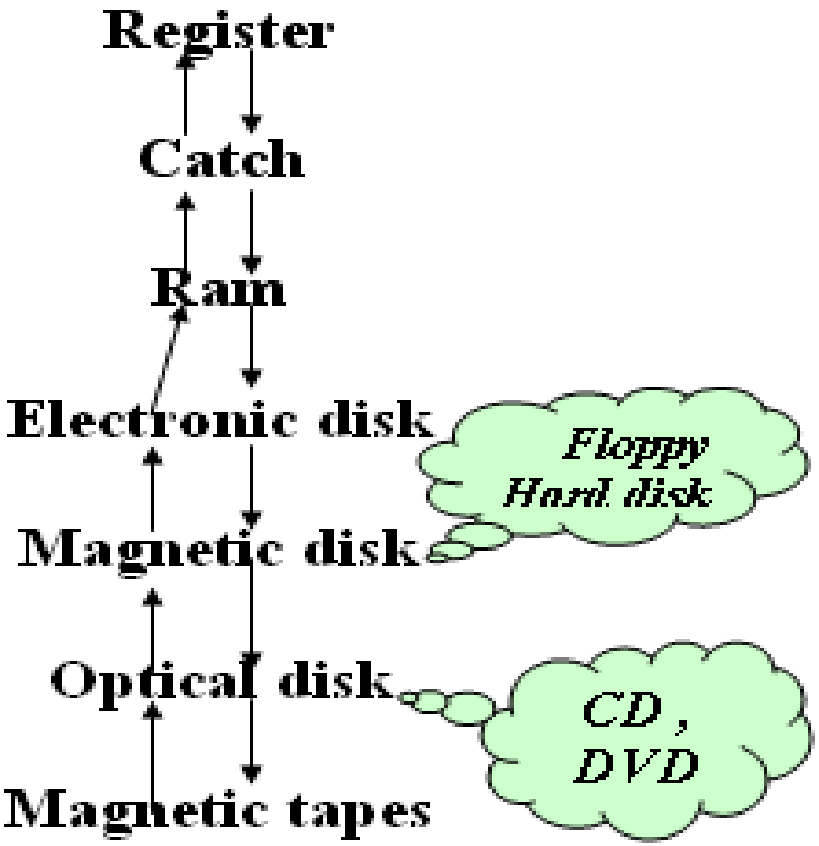
أي تحول لمعلوماته إلى الذاكرة | 1040 | 302

وحسب الشكل التالي:



Storage-Device hierarchy
متسلسلة/جهزة الخزن

وتتكون المخازن الحاسبة وحسب الشكل التالي من اقل مخزن إلى اكبر حجما



هيكلية أجهزة الإدخال والإخراج *I/O structure*

يعد المخزن (Storage) واحد فقط من عدة أنواع أجهزة الإدخال والإخراج المتعددة في الحاسبة يكرس نظام التشغيل على إدارة I/O بسبب الأهمية العالية التي تكمن في هذا الجزء من حيث عملية توفير البيانات وتنفيذها وكذلك بسبب طبيعة تنوع الأجهزة.

يتكون نظام الحاسبة من أجزاء يسمى بالمصادر (Resources) وهي C.P.U و Ram. وكذلك كارتات السيطرة المتعددة (Multi device controllers) والتي تربط جميعا خلال System Bus وكل واحد من كارتات السيطرة للأجهزة بدوره مرتبط بنوع معين من الأجهزة وربما أجهزة السيطرة مرتبط بأكثر من جهاز مثل "USB" Universal Serial Bus ويرتبط بأكثر من "15" جهاز.

يظم أجهزة السيطرة بعض المخازن المؤقتة المحلية (Local Buffer Storage) وكذلك مجموعة من السجلات (Register) الإغراض الخاصة.
وعليه يكون كارتات السيطرة على الجهاز مسؤول عن تحويل البيانات بين الأجهزة المحيطة (الطابعات،الشاشات،.....)
والتي يسيطر عليها وبين المخزن المؤقت له وعموما نظام التشغيل يمتلك مشغلات أجهزة (Device Driver) لكل نوع من Device Controller الجزء البرمجي،أي تكون على شكل سيديات مثل الطابعة لها CD(Device Driver) وهذه الأجزاء البرمجية تشغل هذه الكارتات Device controller have device driver
Print card controller have CD (Device controller)

أي يجعل الجهاز مربوط مع الحاسبة متوائم مع الكارت المسيطر في الحاسبة وكذلك مع نظام التشغيل ***ولبدء عملية I/O***

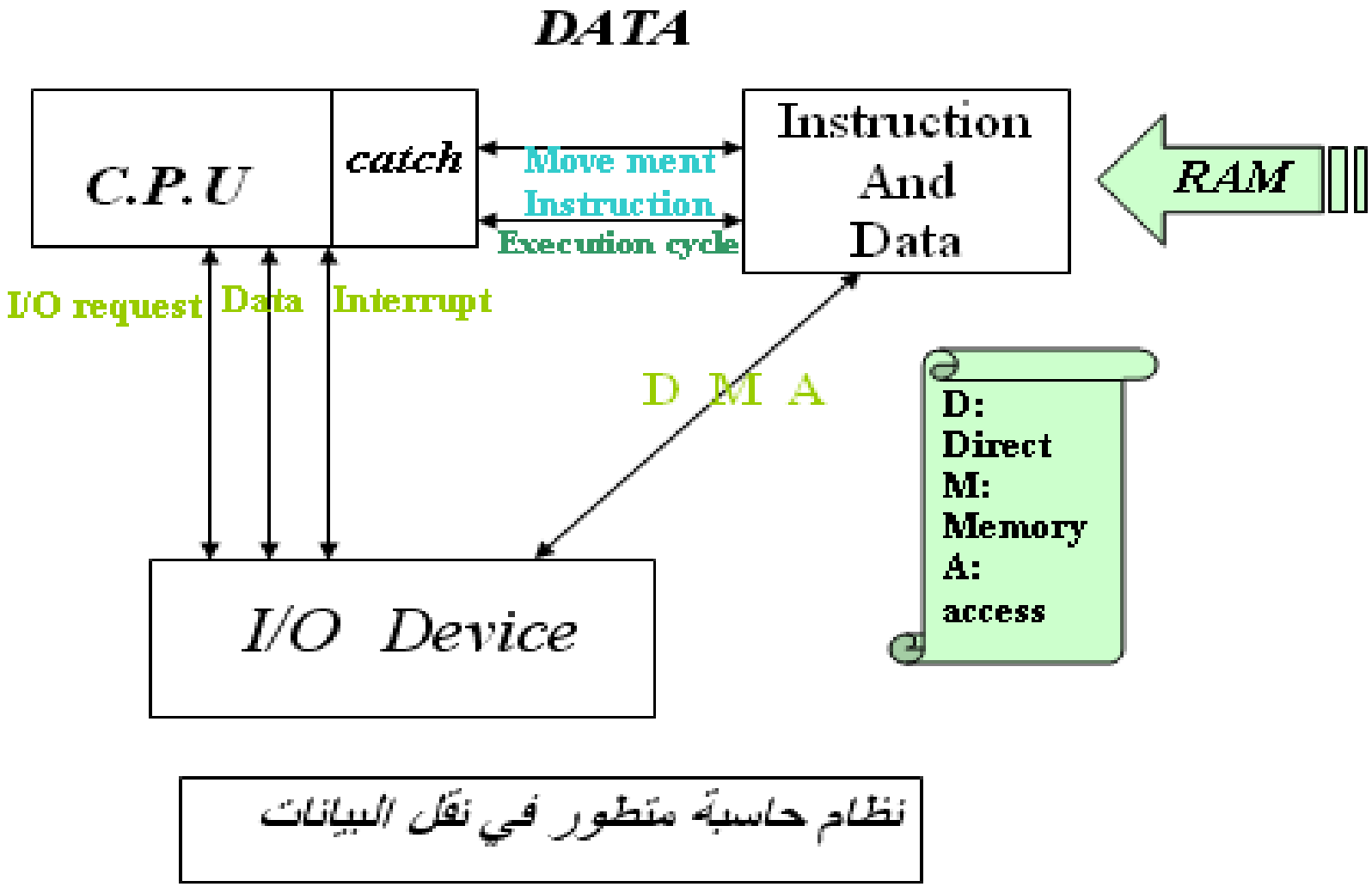
١-يقوم Device Driver بتحميل مسجلات معينة Device Controller .

٢-يقوم Device controller بدوره بفحصه محتويات هذه السجلات وتحديد العمل المطلوب القيام به كقراءة حرف من لوحة المفاتيح.

٣-يقوم بعد ذلك Device controller بالبدء بتحويل البيانات (DATA) من الجهاز إلى المخزن المؤقت المحلي (Local Buffer Storage)وحالة انتهاء نقل البيانات يقوم Device Controller بإخبار Device driver من خلال عملية قطع Interrupt وذلك إن أعطى العملية المكلف بها

وبعد ذلك يقوم بإرجاع السيطرة للـO.S. عملية القطع مناسبة لنقل البيانات القليلة ولكنها غير مقبولة في حالة نقل البيانات كبيرة الحجم.

ولذلك تم استخدام ما يسمى بدخول الذاكرة المباشر DMA
"DIERCT MEMORY ACCESS"
كما بالشكل التالي:



إدارة المخازن
Storage Management

لجعل نظام الحاسبة مناسب للمستفيدين يوفر نظام التشغيل مشاهدة ملائمة ومنطقيا لمخزن المعلومات "Information Storage" يلخص نظام التشغيل من خلال خواص أجهزة الخزن الفيزيائية لديه ليعرفنا بوحدة الخزن المنطقية والتي تسمى بالملف "File" يقوم نظام التشغيل بحفر "MAP" الملفات على وسائط الخزن وكذلك الدخول الى هذه الملفات من خلال أجهزة الخزن (Storage Devices).

إدارة ملف النظام

File_System Management

تعتبر إدارة الملف واحدة من أهم أو أكثر العناصر المرئية"المنظورة" التي يقوم بها نظام التشغيل حيث يستطيع الحاسوب خزن المعلومات (Store Information) على أنواع متعددة من وسائط الخزن(Physical media) ومن وسائط الخزن Hard Disk،القرص الضوئي،الشريط الممغنط ، والتي تعتبر من أكثر وسائط الخزن شيوعا.

كل واحدة من هذه الوسائط "Media" يمتلك مميزات خاصة به. ويسيطر على هذه الوسائط من خلال أجهزة كسائق الأقراص وغيرها وتتضمن هذه الخاصية سرعة الدخول"Access Speed" والسعة"Capacity" وكذلك سرعة نقل البيانات " data-transfer rate" وكذلك طريقة الدخول المستخدمة والتي تكون إما دخول تسلسلي "Sequential" أو دخول عشوائي"Random"

يعتبر الملف File جامع من المعلومات المترابطة "Related" تعرف من خلال المكون لهذه المعلومات.

وبشكل عام يمثل الملف "File" من خلال البرامج

Source And Object Form

وكذلك البيانات وعليه،يمكن أن تصنف ملفات البيانات "data file" حسب مكوناتها أو نوع البيانات التي تظمها هذا الملف تكون عددية "Numeric" و أحـــرف "Alphabetic" أو ثنائيـــة "Binary" وكذلك تكون الملفات من نوع يسمى Free_Form والذي يضم النصوص "Text" أو يكون ذات صيغة ثابتة"Rigidly" كالحقول الثابتة "Fixed Fields" وعليه يكون مفهوم الملف وبشكل عام هو واحد من ما سبق.

يقوم نظام التشغيل بإدارة الملفات من خلال أدارة الأجهزة التي تظم هذه الملفات والمتمثلة بأجهزة الخزن الصلبة " , CD, H/D Floppy Disk"تنظم الملفات ضمن مجلدات تسمى " windx folder" أو دليل في MS-DOS directory.

لكي يسهل استخدامها فعند دخول المستفيدين على الملفات يستطيع المستفيد التحكم على الملفات من خلال القراءة أو الكتابة أو الإضافة على الملف كما في قواعد البيانات أو معالج النصوص يستطيع نظام التشغيل القيام بالفعاليات التالية أثناء إدارة الملفات والمجلدات وهي:

- تكوين وحذف الملفات "FILE"
- تكوين وحذف المجلدات "FOLDER"
- معالجة المجلدات و الملفات
- طبع "حفر" الملفات والمجلدات على المخازن الثانوية
- نسخ المجلدات والملفات على وسائط خزن ثابتة غير متطايرة

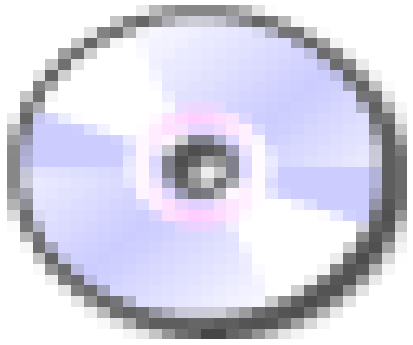
إدارة المخازن الثانوية
Mass_Storage management

وكما رأينا سابقا وبسبب صغر حجم الذاكرة الرئيسية لخزن كل البرامج والبيانات وكذلك بسبب كون هذه الذاكرة متطايرة عند انقطاع التيار الكهربائي "Volatile" يقوم نظام الحاسبة بتوفير مخزن ثانوي" H/D_Secondary Storage" لنسخ البيانات والبرامج الموجودة على الذاكرة على المخزن الثانوي لحفظها عند انقطاع التيار الكهربائي أو عند إطفاء الحاسبة " non volatile".

وعليه عمدت أنظمة الحاسبات المتطورة إلى استخدام الأقراص كواسطة خزن رئيسية "on-line storage" Principe للبرامج والبيانات.معظم البرامج والمتضمنة assemblers, compiler ومعالج النصوص وغيرها و التي تخزن على القرص إلى إن تحمل إلى الذاكرة الرئيسية RAM.

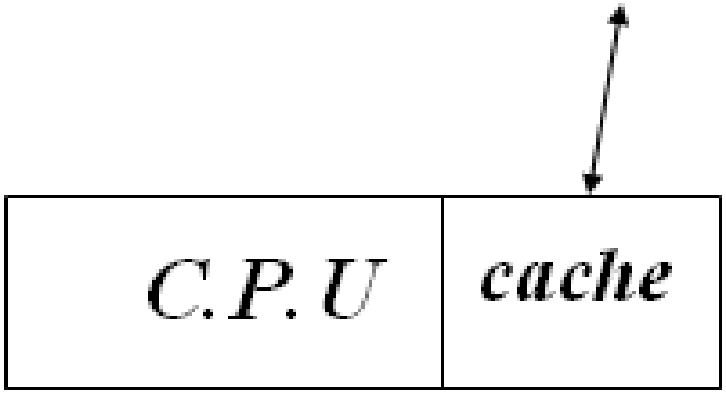
وعليه يستخدم القرص الصلب كمخزن إضافة إلى ذاكرة ثانوية من خلال ما سبق يكون دور نظام التشغيل هو إدارة هذه المخازن من خلال ما يلي:

- إدارة المساحة الفارغة في القرص.
- تحديد المخزن.
- جدولة القرص.



الكاش ميموري
Cache Memory

هي الجزء المهم و الرئيسي في نظام الحاسبة حيث إن المعلومات تحفظ في مخازن الذاكرة كالذاكرة الرئيسية وكذلك تستخدم عادة لحفظ المعلومات حيث يتم نسخ المعلومات في مخزن النظام السريع والمسمى بذاكرة الكاش أو الكاش ميموري وتكون ملاصقة للمعالج وهي تساعد المعالج على جعله مشغول على طول الوقت.



نداءات النظام
SYSTEM CALL

توفر نداءات النظام أسلوب التداخل (Interfaces)إلى الخدمات المتكونة و المتوفرة من قبل نظام التشغيل،أن هذه النداءات تكون بشكل عام متوفرة على شكل برامج فرعية(Sub Routines) مكتوبة بلغة" C و C++" مع التأكد على المهام التي تستخدم لغات المستوى الواطئ Low-level language والتي ربما تقرأ وتكون مكتوبة بتعليمات لغة المستوى الواطئ.

قبل معرفة كيفية تكوين النداءات من قبل نظام التشغيل لنبدأ أولا:

بإعطاء مثال يوضح كيفية استخدام نداءات النظام فمثلا لو أردنا كتابة برنامج مبسط يقوم بقراءة البيانات من ملف معين ونسخ هذه البيانات إلى ملف أخر وخطوات العمل كالتالي:

إن أول إدخال أسماء البرامج "أي ملف الإدخال والذي نقرأ منه البيانات وملف الإخراج الذي نكتب عليه البيانات" هذه الأسماء يمكن أن نعينها بعدة طرق وحسب إمكانية نظام التشغيل وطريقة تصميمه.

احد طرق أو أساليب هي يقوم البرنامج بطلب إدخال اسم ملف الإدخال "Input File"وكذلك إدخال ملف الإخراج " Output file" في الأنظمة المتفاعلة "Interactive system" هذا الأسلوب سوف يتطلب سلسلة من النداءات " Sequence of System call" فيكون كتابة رسالة على الشاشة وبعدها يتم القراءة من لوحة المفاتيح بحروف هذين الاسمين وهذا الأسلوب يتمثل بنظام MS-DOS لكن في الأنظمة التي تعتمد على الفأرة "Mouse-based system" والتي تعتمد كذلك على "ICON"

...ICON-Based system" فتكون القائمة MENU للملفات والتي عادة تعرضها بشكل متسلسل وبسيط كما في نظام Window "API"ضمن هذا الأسلوب يستطيع المستخدم استخدام الفأرة لاختيار اسم الملف المصدر "Source name" وكذلك Windows يمكننا من فتح الملف المقصود "out put file" أو ما يسمى "Destination name" لنضع فيه البيانات هذه المتوالية من الاختيارات تتطلب عدة نداءات من الإدخال والإخراج (I/O system calls)

A P I=Application- Programming- Interface

وعليه يتم الحصول على اسم الملفين أي بمعنى أن البرنامج يقوم بفتح ملف الإدخال"Input File" وتكوين ملف الإخراج" Output File" أن كل هذه العمليات تتطلب نداءات أخرى وكذلك هناك أيضا شروط خطأ محتملة لأي عملية.

عندما يقوم البرنامج بمحاولة فتح الإدخال ربما لا يجد الملف المطلوب عندها يقوم البرنامج بإظهار رسالة على الشاشة يعني أن الملف محمي أو غير مسموح بدخوله وهذه الرسالة هي عبارة عن نداء وعندها ينتهي البرنامج و هذا أيضا نداء أخر.
ففي حالة عدم وجود الملف المطلوب في الحاسبة ونقصد به ملف الإخراج"out put file" وعليه يقوم البرنامج بتكوين ملف جديد باسم معين ففي حالة كان هذا الاسم هو اسم الملف موجود في الحاسبة هذه الحالة ربما تبين حالة إنهاء "abort" وهذه تسمى أيضا نداء أو ربما تحذف الملف الموجود وهذه أيضا نداء وأيضا تكون ملف فهذا يسمى أيضا بنداء كذلك ظهور رسالة تطلب من المستفيد في حالة وجود اسم سابق للملف بإبداء له أو إنهاء البرنامج يسمى بنداء النظام"System Call" أو"Message" .

مفهوم البرنامج "تحت التنفيذ" المنفذ
Process concept

التساؤل الذي فرض نفسه عند مناقشة متطلبات "O.S" هو كيفية استدعاء كل إمكانيات C.P.U حيث إن نظام التشغيل ألحزمي "Batch System" ينفذ Jobs بينما نظام المشاركة الزمنية "Time-shared system" يمتلك برامج المستفيد أو المهام وحتى في نظام التشغيل ذات المستفيد المتفرد "single-user-system" مثلا: Windows يستطيع المستفيد تنفيذ عدة برامج في زمن واحد مثل: "Word" أو عارض الصفحة "Web Browser" "e-mail package " وعليه حتى إذا كان المستفيد يستطيع تنفيذ برنامج واحد فقط في كل مرة يحتاج نظام التشغيل ربما إلى مساندة الفعاليات البرمجية الداخلية والتي يمتلكها مثال ذلك إدارة الذاكرة.

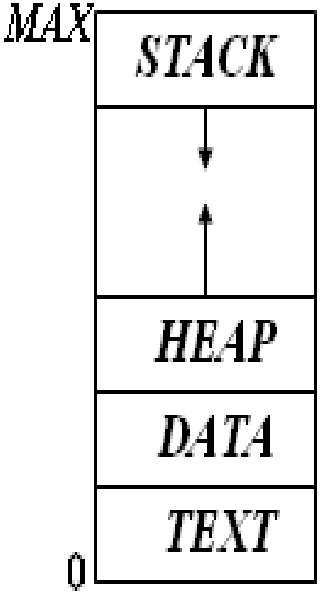
وفي عدة أوجه تكون هذه الفعاليات البرامجية متشابهة وعليه يستطيع أن يستدعي كل هذه البرامج التي تكون تحت التنفيذ أو مايسمى Process.والـProcess هو البرنامج أو الـJob الذي ينفذ من قبل الـC.P.U.

إن مصطلح الـJob و Process يستخدمان في الاغلب بشكل متبادل في هذا الموضوع وعلى الرغم من إن المصطلح المفضل هو Process ولكن في كثير من مصطلحات ونظريات أنظمة التشغيل يتم استخدام مصطلح Job وهي ممكن إن تؤدي إلى تظليلنا في تجنب استخدام المصطلح الشائع الاستخدام و المتضمن بكلمة Job مثال ذلك:مصطلح "Job Scheduling" ولكن مصطلح "Process" هو بديل لـJob.

البرنامج تحت التنفيذ
The process

بشكل عام أن Process في برنامج تحت التنفيذ وProcess هي أكثر من أن تكون برنامج أو شفرة برنامج و التي تعرض في بعض الأحيان بشكل مقطع نصي . وProcess تتضمن الفعاليات الحالية والتي تتمثل من خلال قيمة عداد البرنامج ومحتويات مسجلات المعالج.بشكل عام وProcess كذلك تظم "Process Stack" والذي يظم بيانات وقتية "temporary data" مثلا: معالج التطبيق Function Parameters "Local Variables address" و كذلك "Data section" والذي يظم "Global Variables" و Process يظم أيضا Heap "مكدس" والذي هو ذاكرة تحدد بشكل إلي خلال التنفيذ Process وعليه يمكن تمثيل هيكلية Process بالشكل التالي:

Process in Memory



يمكن التأكد على ان البرنامج نفسه هو ليس Process يمثل البرنامج كائن غير مفعل "Passive entity" والذي يظم قائمة من التعليمات المخزونة على القرص والتي غالبا ما يدعى بالملف التنفيذي "Executable entity" بعداد البرنامج والذي يحدد التعليمة القادمة للتنفيذ ومجموعة من المصادر المشتركة .وعليه يصبح البرنامج Process هو عبارة عن كائن مفعل " Active entity" بعداد البرنامج والذي يحدد التعليمة القادمة للتنفيذ ومجموعة من المصادر المشتركة.وعليه يصبح البرنامج process عندما يحمل "Executable file" إلى الذاكرة.هناك طريقتين لتحميل "Executable file":أولا:الضغط المزدوج على الإيقونة التي تمثل"Executable file"إدخال اسم الملف " Executable file" على"Command Line" كما في المثال التالي:

C:\> prog.exe ↵

حالة البرنامج تحت التنفيذ
Process State

عند التنفيذ Process تنفذ حالاتها "States" وتعرف حالة Process من خلال الفعاليات الحالية وكل Process ربما هي واحدة من الحالات الآتية :

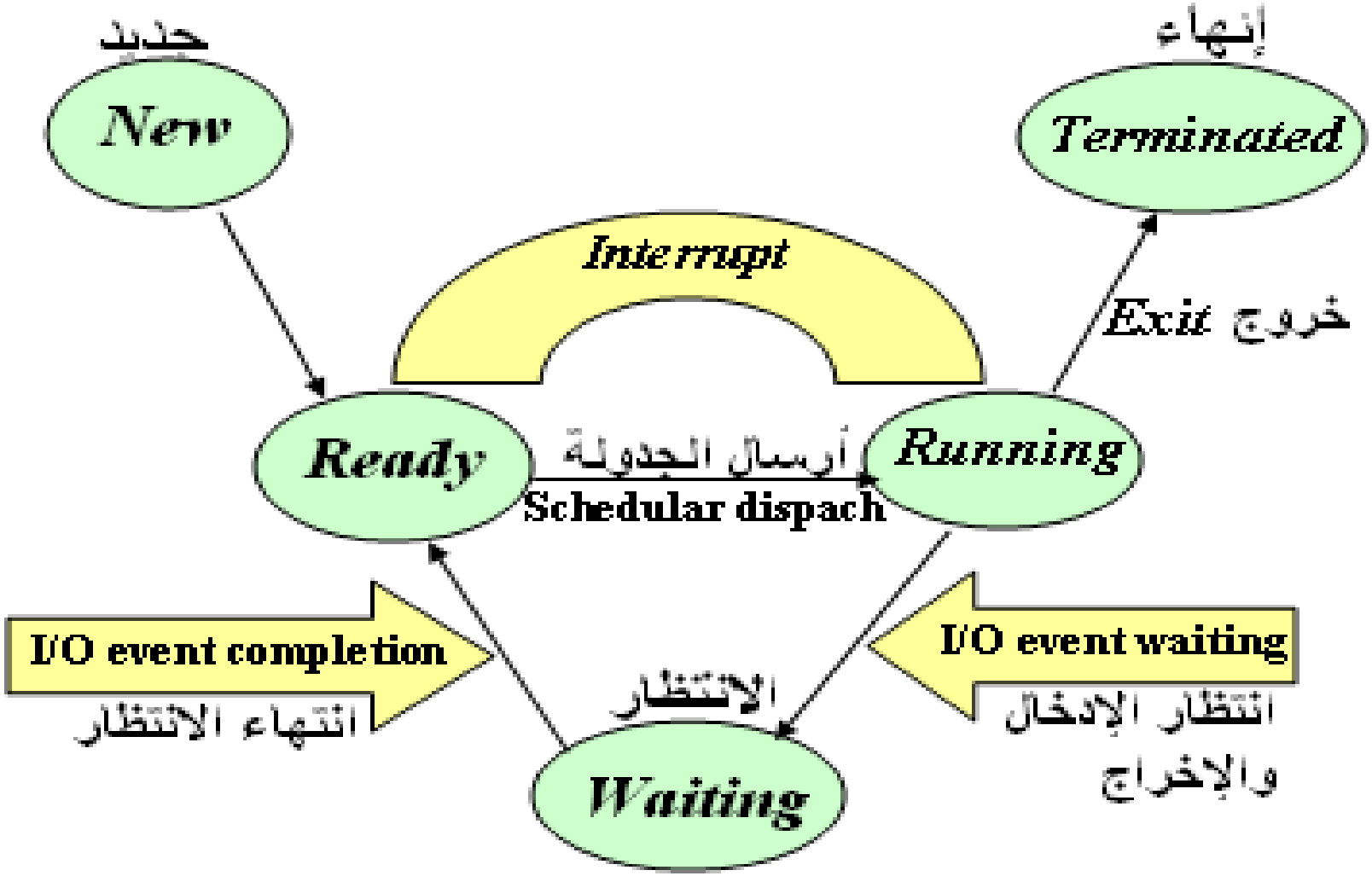
"New": Process تكوين

"Running": Process تنفيذ تعليمات

"Waiting": تدخل Process حالة الانتظار لحدوث عدم توفر المكتبات المطلوبة من قبلها وبذلك لاستغلالها من قبل Process أخرى

"Ready": انتظار Process للتنفيذ من قبل المعالج C.P.U "جاهزة أو مستعدة للتنفيذ من قبل C.P.U"

الشكل التالي يوضح حالات Process


جديد
New
إنهاء
Terminated
Interrupt
Exit خروج
Ready
أرسال الجدولة
Schedular dispach
Running
I/O event completion
انتهاء الانتظار
الانتظار
Waiting
I/O event waiting
انتظار الإدخال
والإخراج

قطاع السيطرة للـProcess

Process Control Block

تمثل كل Process في نظام التشغيل من خلال
Process Control Block
"P.C.B" وكذلك يدعى بـ"Task Control Block" "T.C.B"
والشكل التالي يمثل الـ P.C.B :

Process State
Process Number
Program Counter
Registers
Memory Limits
List of Open File
……

يظم Process عدد من المقاطع المعلوماتية والمشتركة مع Process معينة التي تظم الأشياء الآتية:

1- Process State

حالة Process والتي تكون إما New أو Waiting أو Interrupt

2- Program Counter

يشير إلى عنوان التعليمة القادمة للتنفيذ .

3- C.P.U Register

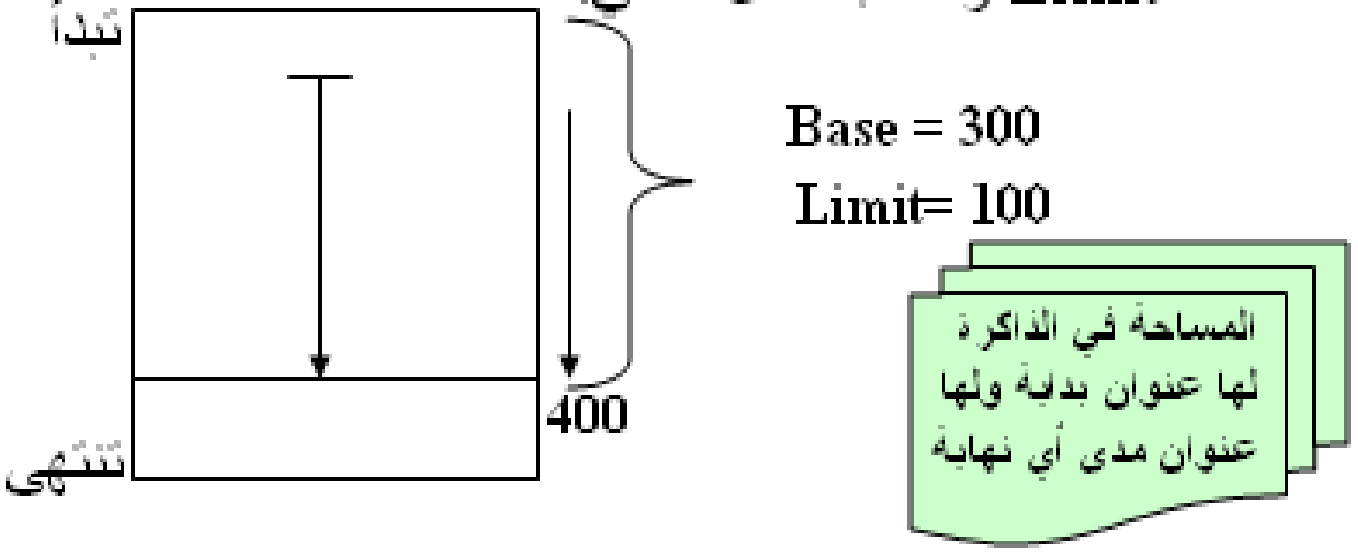
تكون المسجلات متنوعة من حيث النوع والعدد وهي تعتمد على معمارية الحاسبة والتي تظم الخزن التراكمي "Acc" والفهرسة.

4- C.P.U Scheduling Information

تضم هذه المعلومات أفضلية Process والمؤشر إلى جدولة الطابور "Scheduling queue".

5- Memory- Management Information

ربما يضم معلومات القيم للمسجلات أي الأساس Base والى المدى Limit وكما بالشكل التالي:



6- Accounting Information

تظم كمية المعلومات الخاصة بـC.P.U و Real-Time المستخدم وكذلك قائمة الملفات المفتوحة.

جدولة البروسس
Process Scheduling

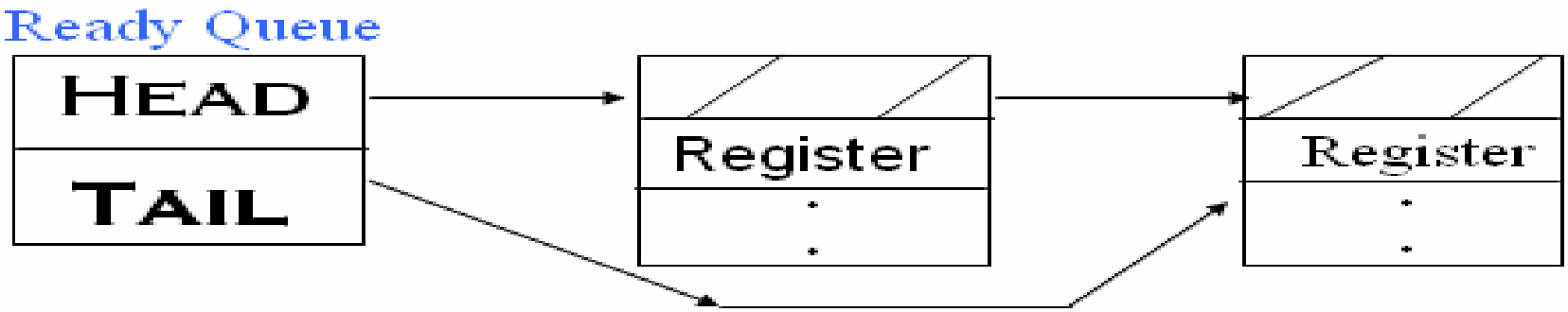
إن هدف البرمجة المتعددة "Multi Programming" هو تنفيذ عدد من البروسس "Process" في كل الأوقات، وكذلك لزيادة استخدام C.P.U . ولكن هدف المشاركة الزمنية "Time Sharing" هو تحويل "Switch" الـ C.P.U بين البروسس وبشكل متكرر بحيث يحس المستخدم وكأن الـ C.P.U مخصص له بحيث يتفاعل"Process" مع كل برنامج عند التنفيذ ولتحقيق هذه الأهداف تقوم بجدولة البروسس "Process Scheduling" وباختيار واحد من البروسس المتواجد ولتنفيذها على C.P.U ولكن في الأنظمة ذات نظام المعالج المنفرد فتكون العملية مختلفة أي سوف لا يكون أكثر من بروسس واحد للتنفيذ والباقي من البروسس سوف ينتظر حتى يفرغ C.P.U ويعاد جدولتها.

جـــدولة الطوابير
Scheduling Queue

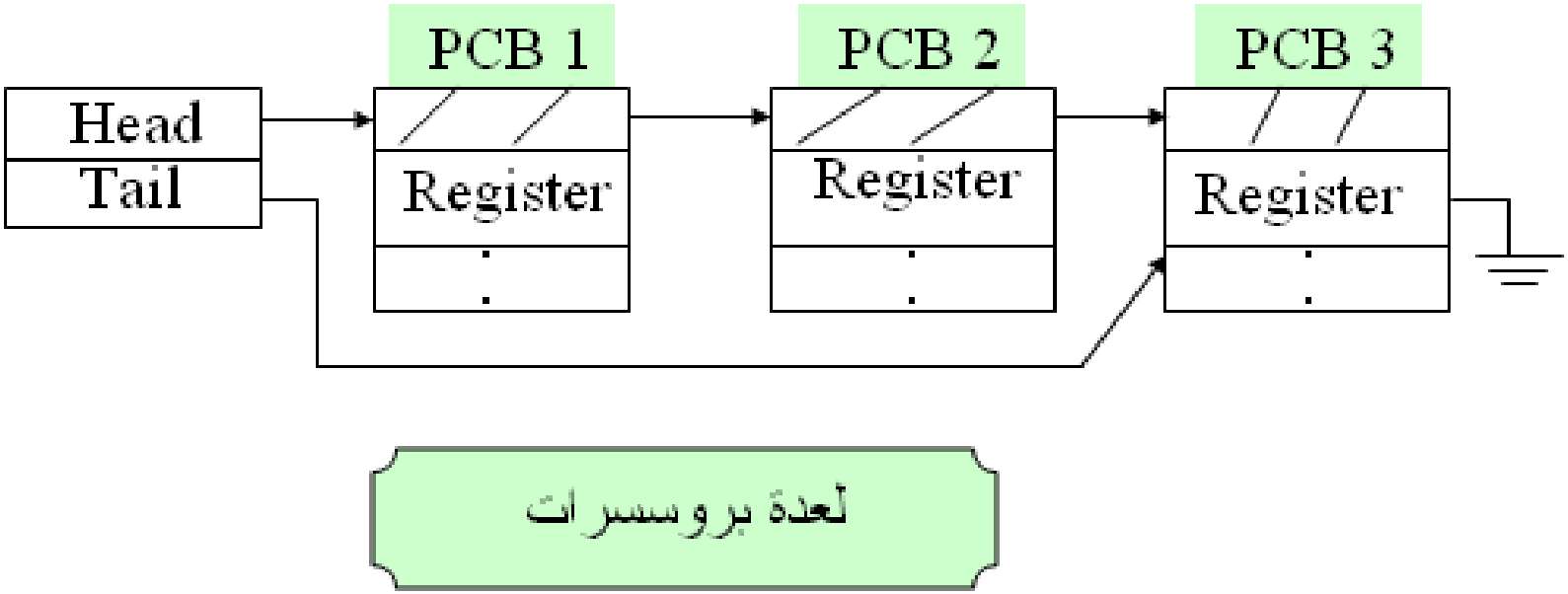
عندما تدخل البروسس إلى النظام "نظام الحاسبة" سوف توضع هذه البروسس في "Job Queue" والذي يظم كل البروسس في النظام.
إن البروسس التي تكون في الذاكرة الرئيسية "Main Memory" وهي عادة تكون مستعدة و منتظرة للعملية، التنفيذ عادة يحفظ في قائمة تسمى Ready Queue "طابور الاستعداد للتنفيذ".
هذه Queue "الطابور" يكون مخزون عادة على شكل " Linked List" قائمة مترابطة أو موصولة يظم عنوان (Ready-Queue) مؤشرات "Pointers" تشير إلى البداية"First" أو P.C.B ونهاية أخر P.C.B في القائمة List .
كل P.C.B يظم حقل مؤشر يشير إلى P.C.B. القادمة من Ready Queue .

يظم نظام الحاسبة طوابير من "Other Queue" عندنا تحدد البروسس إلى C.P.U يقوم بتنفيذها وعند إتمام التنفيذ تخرج أو تقطع "Interrupt" أو ينتظر "Wait" بسبب احتياجها إلى I/O المستخدم من قبل بروسس أخرى.

نفرض إنها تطلب I/O أي البروسس يستخدم جهاز معين مثلا القرص، وربما هذا القرص مستخدم من قبل بروسس أخرى فتضطر هذه البروسس إلى الانتظار لحين توفر القرص لاستخدامه تسمى القائمة التي تظم هذه البروسس المنتظر"Device queue" طابور الجهاز والشكل التالي يوضح طابور البروسس.



طابور الاستعداد لبروسسين



لعدة بروسسرات

زمن التنفيذ للبروسس من قبل C.P.U
"C.P.U – BURST TIME"

الخوارزمية

(S.J.F)Shortest-Job-First Scheduling

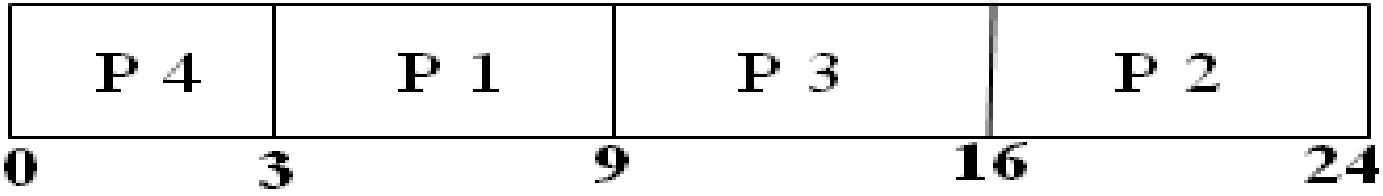
هذه الخوارزمية تجدول اولا بغير زمن البروسس هذا الاساس في تحديد البروسس التي تنفذ اولا أي البروسس ذات الزمن الاقصر "زمن تنفيذها" فالأقصر اولا ثم التي تليها وهكذا.

مثال:باستخدام خوارزمية**(S.J.F scheduling)** وبزمن تنفيذ **"Length c.p.u burst"**ملي سكند

Process(p1,p2,p3,p4)

Burst time(6,4,3,1)على التوالي

الحل:حسب خوارزمية S.J.F وباستخدام مخطط "Gantt Chart" ونختار الأقصر



زمن الانتظار p1= 3 و P2=16 و P3=9 و P4= 0

Average Waiting Time=(3+16+9+0)/4=7 M.S

DISPATCHER*

يعتبر حد العناصر المطلوبة في c.p.u-scheduling ويعرف على أن النموذج الذي يعطي سيطرة C.P.U إلى البروسس المختارة للتنفيذ.

جـــــدوليه الأفضـــــلية
Priority Scheduling

خوارزمية S.J.F هي حالة خاصة من هذه الخوارزمية الأفضلية "Priority" متصلة مع كل بروسس وبعدها يحدد C.P.U لهذه البروسس مع أفضلية عليا "High Priority" لكن عندما تتساوى الأفضلية فتستخدم خوارزمية(F.C.F.S) ونستخدم خوارزمية الأفضلية الأرقام الدنيا "Low number" للأفضلية العليا و الأرقام الكبيرة (للأفضلية الدنيا) و كما في المثال الأتي:

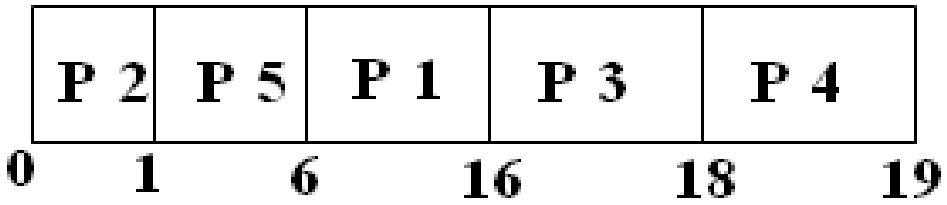
Consider the following set of processes assumed to have arrived at time 0, in the order P1,P2….,P5 ,
with the length of c.p.u burst given in millisecond

Process(P1, P2 ,P3 ,P4 ,P5)

Burst time(10,1,2,1,5)

Priority(3,1,4,5,2)

using priority scheduling we would scheduling this process according to the following Gantt Chart.



Average waiting Time=(0+1+6+16+18)/5= 8.2 M.S

PREEMPTION الأولويـــــة

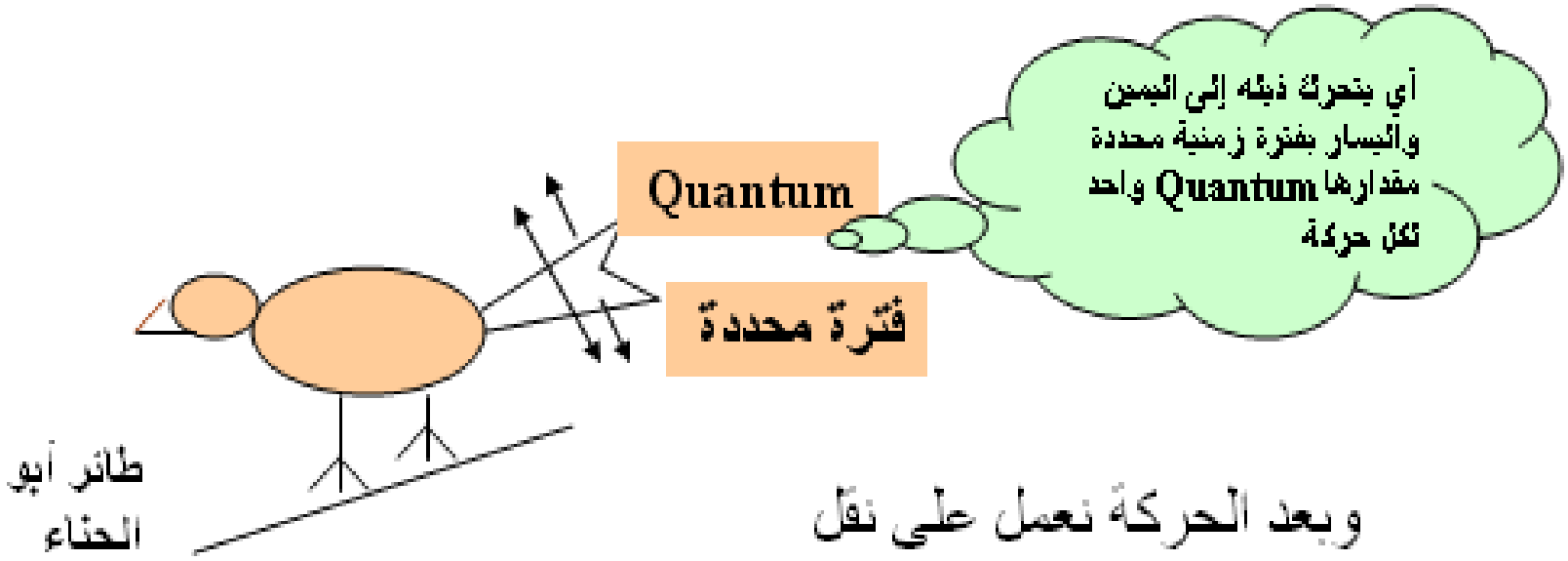
(R.R) Round- Robin Scheduling

جدولة دورة طائر الحناء

إن خوارزمية أبو الحناء صممت خصيصا للأنظمة ذات المشاركة الزمنية "Time Sharing System" وهي مشابهة لخوارزمية "F.C.F.S" و أضيفت الأولوية للتحويل "Switch"بين البروسس استخدمت وحدة الزمن الصغيرة تسمى "Time-Quantum" أو Time-Slice كحركة الطير بذيله و Time Quantum بشكل عام يكون من 10-100ملي سكند .

يعامل أو يعالج طابور الاستعداد "Ready queue" على أن طابور دائري "Circular queue" ويدير c.p.u scheduling Ready queue ويحدد c.p.u لكل بروسس ولفترة زمنية محددة بـOne quantum ولتطبيق خوارزمية أبو الحناء "R.R" يجعل Ready queue آو تعامل Ready queue على أساس F.I.F.O "First In First Out"-(F.I.F.O-Queue)من البروسس.

و البروسس الجديدة تضاف إلى ذيل"Tail" طابور الاستعداد
مجدول الـC.P.U (C.P.U Scheduler) ينتقي "Picks" أول
بروسس من Ready queue، نعمل Interrupt قطع للمؤقت
"Timer" بعد Time Quantum
"Sets a timer to interrupt" في " After 1 time
quantum" كما في الرسم التالي:



وبعد الحركة نعمل على نقل
One quantum dispatches للبروسس

وعليه وضمن هذه الحركة هناك شي واحد يحدث ما بين شيئين إما البروسس يحتاج البروسس C.P.U burst اقل من quantum واحد ومن هذه الحالة البروسس نفسها سوف تترك أو تحرر C.P.U من سيطرتها طواعية "Voluntarily" . وعليه الجدول سوف يقدم البروسس القادمة في Ready queue للتنفيذ ويعطيها صلاحية السيطرة على C.P.U وإلا."الحالة الثانية" عندما يكون يحتاج أكثر من quantum سوف يعمل على قطع "Interrupt" للـO.S وسياق التحويل سوف يعمل على تنفيذ و البروسس سوف تذهب إلى ذيل Ready queue كبروسس جديدة وسوف يقوم المجدول بتخصيص البروسس القادمة في Ready queue وعليه سوف يكون تعريفه معدل زمن الانتظار غالبا طويلا .

وكما في المثال التالي:

Consider the following set of processes that arrive at time 0, with the length of the C.P.U burst given in millisecond:

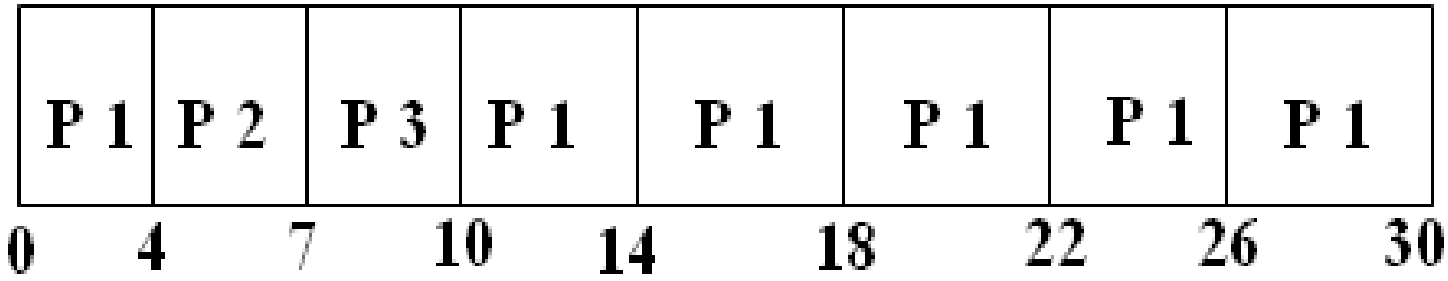
Process(P1 ,P2 ,P3)

Burst time(24 ,3 ,3)

If we use a time quantum =4 millisecond the process P1 gets the first 4 millisecond.

يبقى لها 20 millisecond وهي تأخذ الأولوية بعد تنفيذ أول quantum وتذهب إلى الذيل بعدها سيقوم C.P.U يقوم بتنفيذ P2 "Next process" في طابور الاستعداد P2 بزمن تنفيذها اقل من الزمن المحدد أي quantum 4 فتنفذ وتخرج قبل انتهاء الفترة المحدد .

بعدها ينفذ الـP.U.C P3 وهي أيضا سوف تنفذ قبل انتهاء الوقت المحدد(4quantum) بعدها سيعود C.P.U إلى P1 والكوانتوم إضافي وكما يأتي كمخطط Gantt Chart.



Average Waiting Time=4+7+(P1(10-4))/3
=17/3 = 5.66 Millisecond

R.R البروسس لا تحدد C.P.U أكثر من Quantum واحد

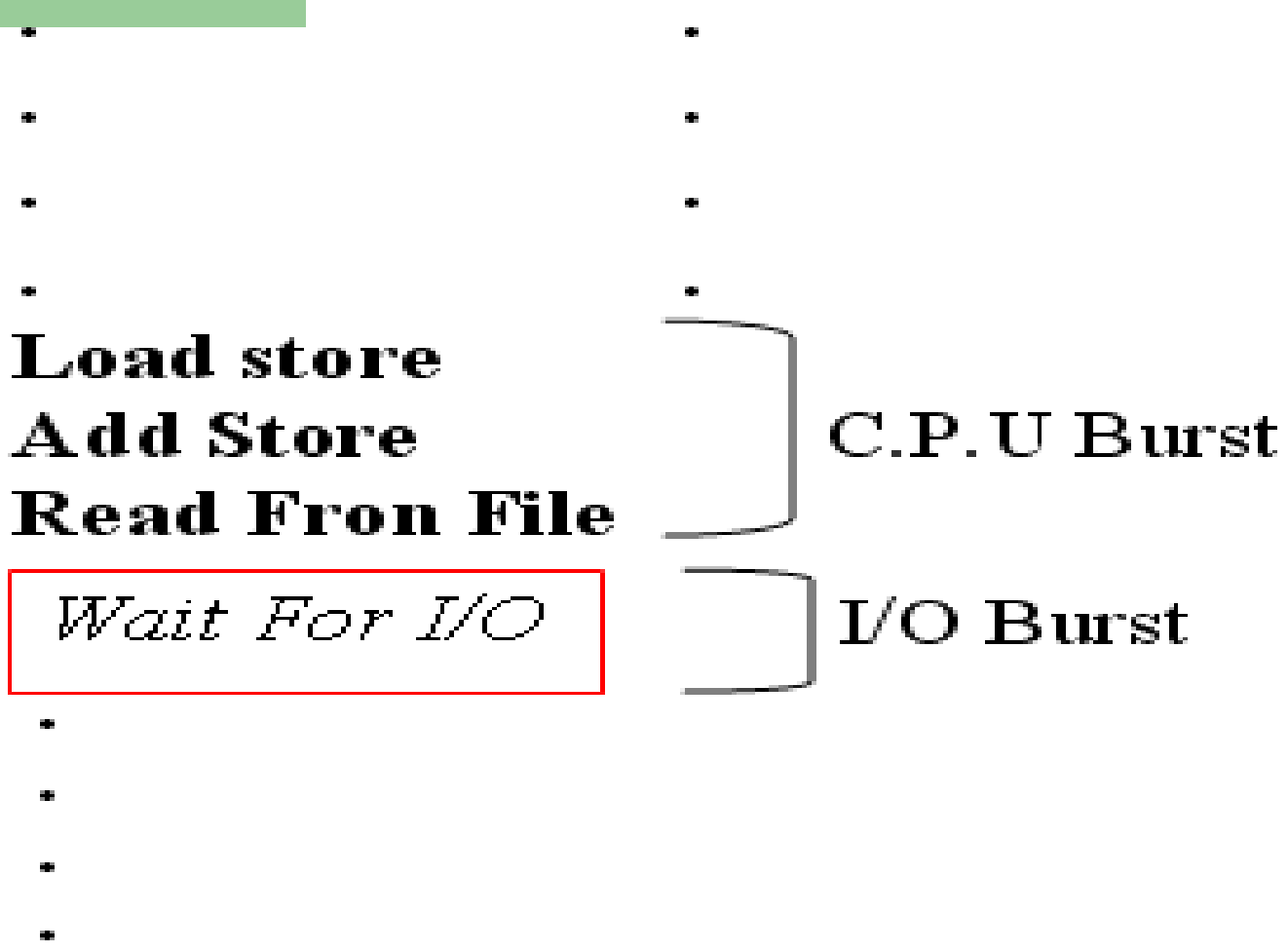
جدولة C.P.U
C.P.U SCHEDULING

تعتبر جدولة المعالج هي الأساس الذي تعتمد عليه أنظمة التشغيل متعددة البرمجة . وذلك بالتحول بين عدد من البروسس "Process" يستطيع نظام التشغيل جعل الحاسوب أكثر فعالية وإنتاجية من خلال هذه الخاصية.

*دورة اندفاع "انفجار" إدخال وإخراج C.P.U

C.P.U – I/O Burst Cycle

نجاح جدولة المعالج للبروسس يعتمد على خاصية المشاهدة التي تمتلكها البروسس أي أن تنفيذ Process يتضمن دورة من تنفيذ C.P.U و انتظار الإدخال والإخراج (I/O Wait) تتناوب البروسس بين حالتي (C.P.U-Execution) تنفذ C.P.U وحالة (I/O Wait) انتظار الإدخال و الإخراج وهذا يسمى بالعملية الانفجارية (C.P.U) وأخيرا انفجار واندفاع C.P.U " C.P.U Burst" ينتهي من النظام عندما يطلب النظام إنهاء تنفيذ البروسس وكما في الشكل التالي:



I/O Burst

سلسلة التناوب للـC.P.U

مجدول C.P.U
C.P.U Scheduler

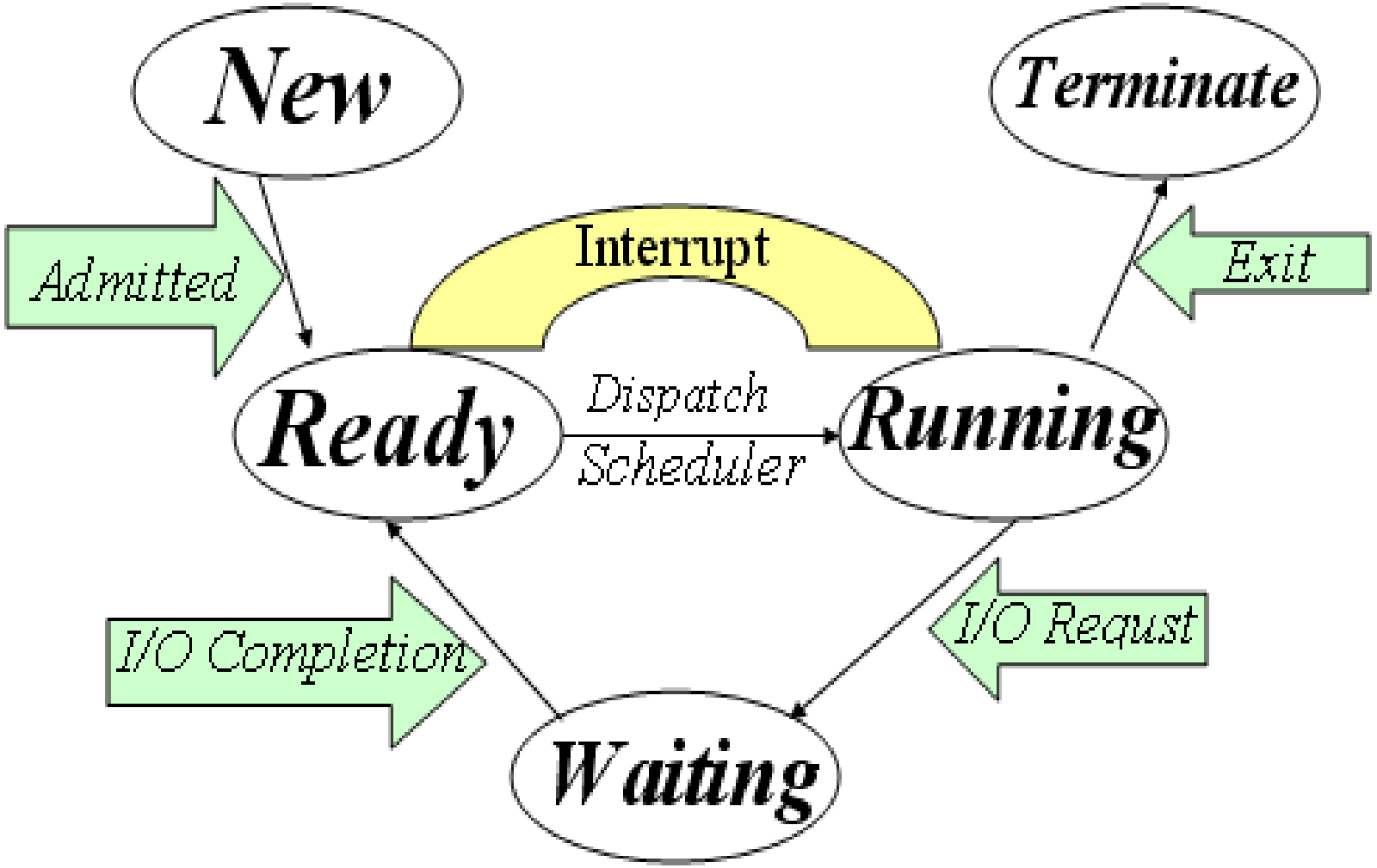
عندما يكون أو يصبح C.P.U بدن عمل "Idle" نظام التشغيل يعمل بإلحاح على اختيار واحدة من البروسس المتوفرة في الذاكرة ضمن ما يسمى Ready –Queue "طابور الاستعداد" للتنفيذ هذه البروسس .

عليه اختيار العملية (Process) تحمل أو تنجز بمصطلح يسمى Short- Term- Scheduler أو ما يسمى C.P.U Scheduler يقوم المجدول باختيار البروسس من البروسسرات أو العمليات المتواجدة في الذاكرة الرئيسية و التي تكون مستعدة للتنفيذ من قبل C.P.U . السجل "القيد" هو P.C.B للبروسس "العمليات" والتي تكون متواجدة ضمن Ready queue ضمن الذاكرة مستعدة للتنفيذ.

جدوليه الأولوية
Preemptive Scheduling

قرارات جدولة C.P.U ربما يستمر تحت الظروف التالية :

١. عند تحول البروسس من حالة Running إلى حالة Waiting "نتيجة لطلب I/O".
٢. عندما تتحول البروسس من حالة Running إلى حالة Ready بسبب Interrupt.
٣. عندما تتحول البروسس من حالة Waiting إلى حالة Ready "إكمال الحصول على I/O".
٤. عندما ينتهي تنفيذ البروسس "Terminate" والشكل السابق و المسمى بحالة البروسس يوضح هذا التقاطع.


New
Terminate
Admitted
Interrupt
Exit
Ready
Dispatch
Scheduler
Running
I/O Completion
I/O Requst
Waiting

خوارزمية الجدولة
Scheduling Algorithm

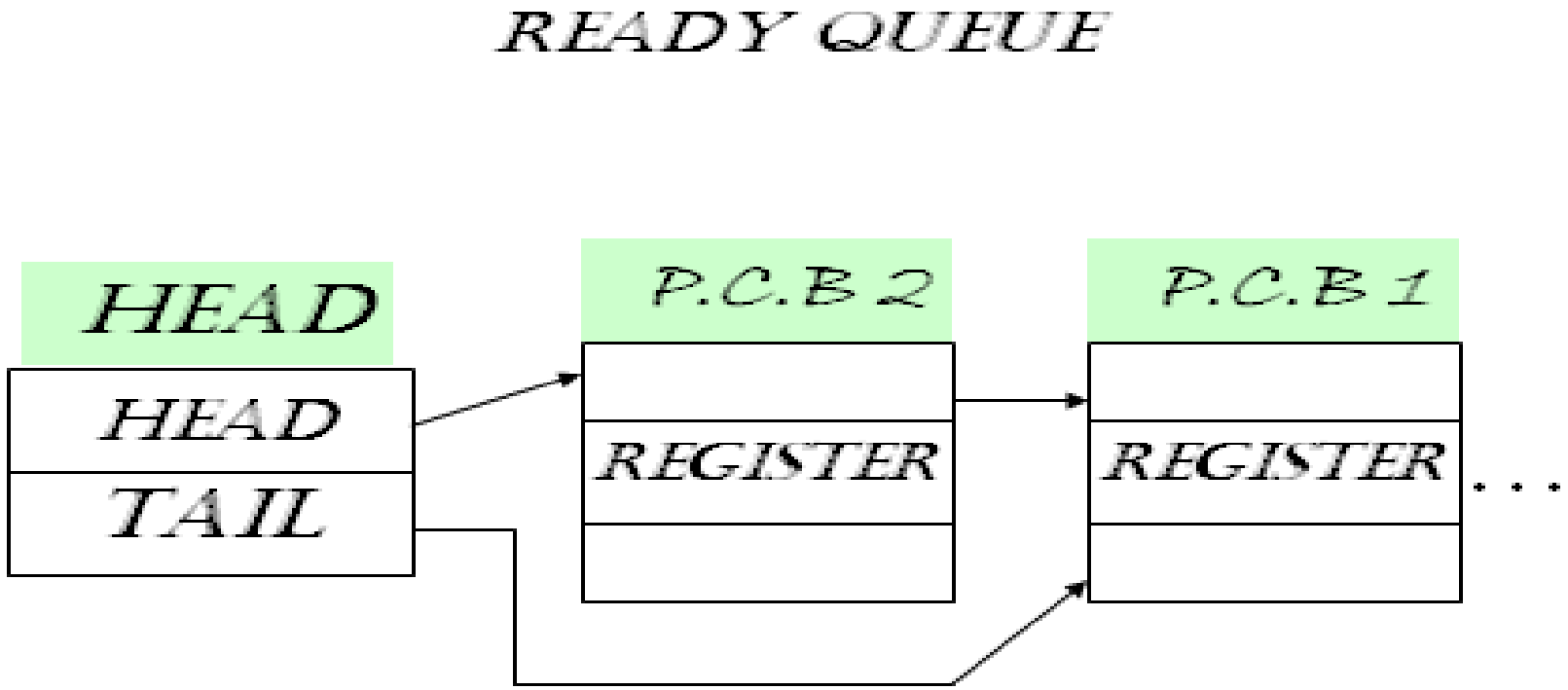
إن جدولة C.P.U تتناول مشكلة القرار في اختيار البروسس من بين البروسسرات المتواجد في Ready Queue وتحديدها إلى C.P.U وعليه هناك عدد من الخوارزميات المختلفة لجدولة C.P.U بعض هذه الخوارزميات :

1-(F.C.F.S)First-Come-First-Served Scheduling

إن جدولة الذي يأتي أولا يخدم أولا أو خوارزمية "F.C.F.S" وعليه فان البروسسرات تطلب C.P.U أولا تحدد للتنفيذ أولا من قبل C.P.U الطريقة"F.C.F.S Policy"تدار بسهولة مع طابور " F.I.F.O queue" .

عندما تدخل البروسس إلى Ready queue والتي تكون ضمن قائمة "Linked List" تكون ما يسمى P.C.B والتي تربط إلى ذيل الطابور.

وعليه عندما يكون C.P.U حر "Free" يحدد البروسس أو C.P.Uإلى البروسس عند رأس Queue، "Head Queue" .
كما في الشكل التالي:



الحل

من خلال استخدام خوارزمية F.C.F.S وباستخدام مخطط جانت "Gantt Chart" .

ملاحظة: مخطط جانت Gantt يستخدم لتحديد زمن كل بروسس وبالتتابع حسب دور كل واحدة أو تسلسل كل واحدة ويكون بالشكل التالي:

حيث يؤشر الرأس إلى أول بروسس دخلت والتي ضمت عملية Linked List ستكون الأخير و البروسس الأخيرة ستكون الأولى

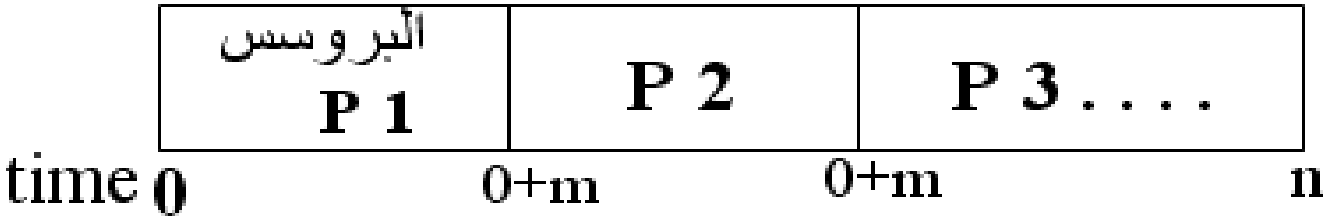
تأمل المثال التالي:

لو كان لدينا عدد من البروسس زمن التنفيذ أو ما يسمى بزمن الانفجار "Burst Time" لكل واحدة كالأتي:

Process (P1 ,P2 ,P3)

Burst Time (24 ,3 ,3)

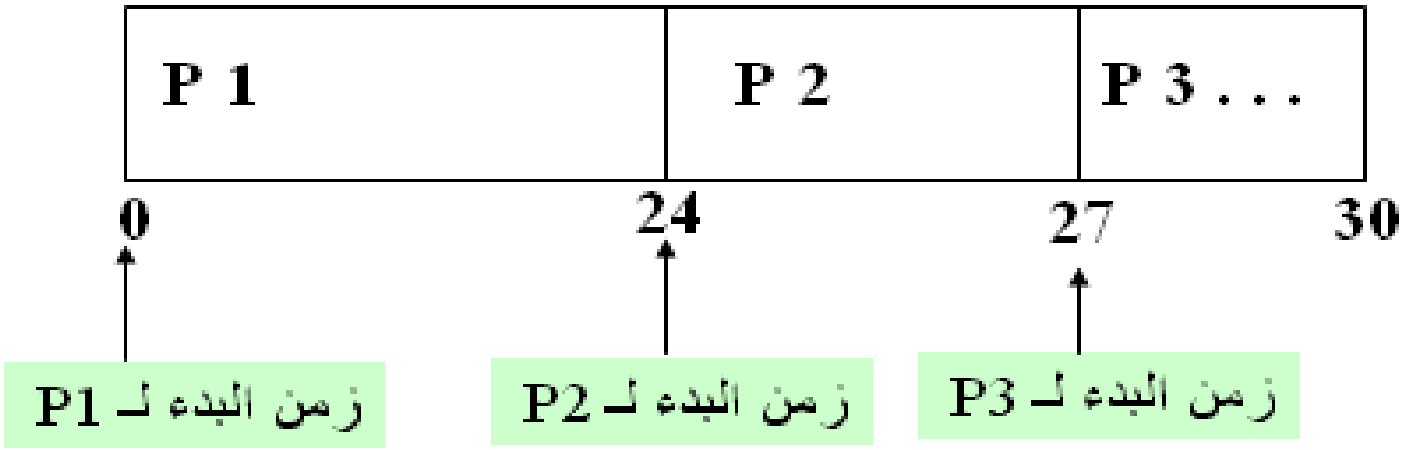
و إن زمن وصول هذه البروسس هو صفر مع C.P.U Burst مع اندفاع و الانفجار C.P.U Burst Time يكون معدل زمن الانتظار "لهذه الطريقة ذا طول مناسب".



تكون أول بروسس جاء تخدم أولا أي تنفذ أولا فيكون

Time: P1=24 ⟶ P2=3+4=27 ⟶ P3=27+3=30

وعليه يكون



A.W.T=(0+24+27)/3 = 51/3 = 17 Millisecond

حيث يأخذ زمن البدء لكل بروسس وكم موضح في الرسم

أمثلة عن نظام التشغيل
Operating System Example

سوف نتطرق إلى إن وصف تعريفات المجدولات "Scheduling Policies" لكل من Solaris , Windows , Sp1 , Sp2 , Vista , Linux

ومن الأمثلة على مجدول الـSolaris وهي:

1— Real Time
2— System
3— Time Sharing
4— Interactive

أمثلة على Windows XP :

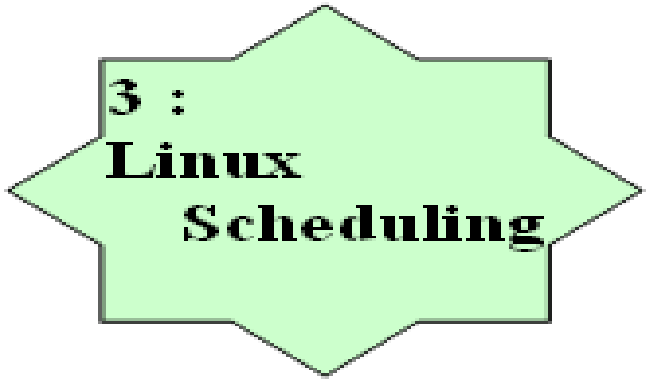
Priority-Based ,Preemptive Scheduling Algorithm

وعليه فان Win 32 API

Windows 32 Application Programming Interface

نحدد عدة فئات للأولوية والتي بعدها تحدد البروسس لأي من هذه الفئات.

1-Real Time – Priority Class.
2-High – Priority Class .
3-Above – Normal – Priority Class.
4-Normal – Priority Class.
5-Below – Normal Priority Class.
6-Idle – Priority Class.



يعتمد على خوارزمية مجدولة Unix وهو نظام قديم استخدم حاسبات Mini Computer – Main Frame

جدولة الطابور المتعدد المستويات

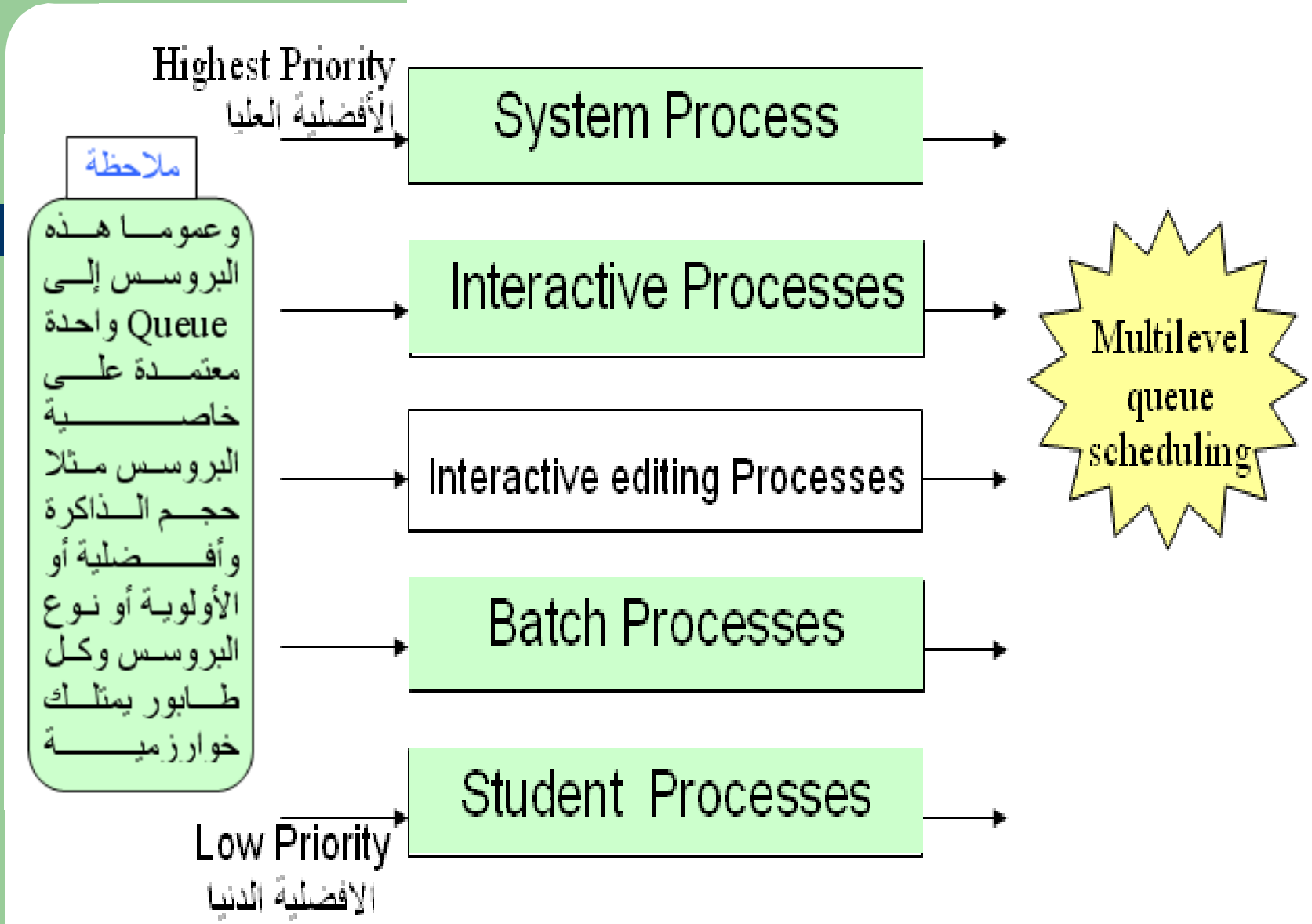
Multi Level Queue Scheduling

نوع أخر من خوارزميات الجدولة والتي كونت للبروسس و التي صنفت ببساطة إلى مجموعتين و هما:

١ ـ المجموعة المتفاعلة "Interactive" وتسمى"Forground"

٢ ـالمجموعة الحزمية"Batch" والتي تسمى"Back Ground"

أي الخلفية (Back Ground) والواجهة (Forground) وهذه المجموعتين تملكان متطلبات زمن استجابة مختلف (Response-Time) وعليه فان البروسسرات الخاصة بالواجهة "Forground Process" تمتلك أفضلية "priority" أعلى من "Back Ground Processes" وعليه فان Multilevel queue Scheduling Algorithm تجزأ أو تقسم (Partitions) الـ Ready queue إلى عدة طوابير مختلفة وكما بالشكل التالي :


Highest Priority
الأفضلية العليا
System Process
ملاحظة
وعموما هذه البروسس إلى Queue واحدة معتمدة على خاصية البروسس مثلا حجم الذاكرة وأفضلية أو الأولوية أو نوع البروسس وكل طابور يمتلك خوارزمية
Interactive Processes
Interactive editing Processes
Batch Processes
Student Processes
Low Priority
الافضلية الدنيا
Multilevel queue scheduling

تقييم الخوارزمية
Algorithm Evaluation

هنالك عدد من خوارزميات الجدولة . و كل واحدة تمتلك معالج خاص بها و إن عملية اختيار المعالج يكون أمر بالغ الأهمية وعليه يمكن أن نضع معيار يمكن إن يعتمد عليه في اختيار الخوارزمية وهذا المقياس أو المعيار يسمى "بزمن الاستجابة" Response Time أو يسمى C.P.U Utilizations وهذا المعيار أو المقياس يشمل على بعض الضوابط وهي كالأتي:

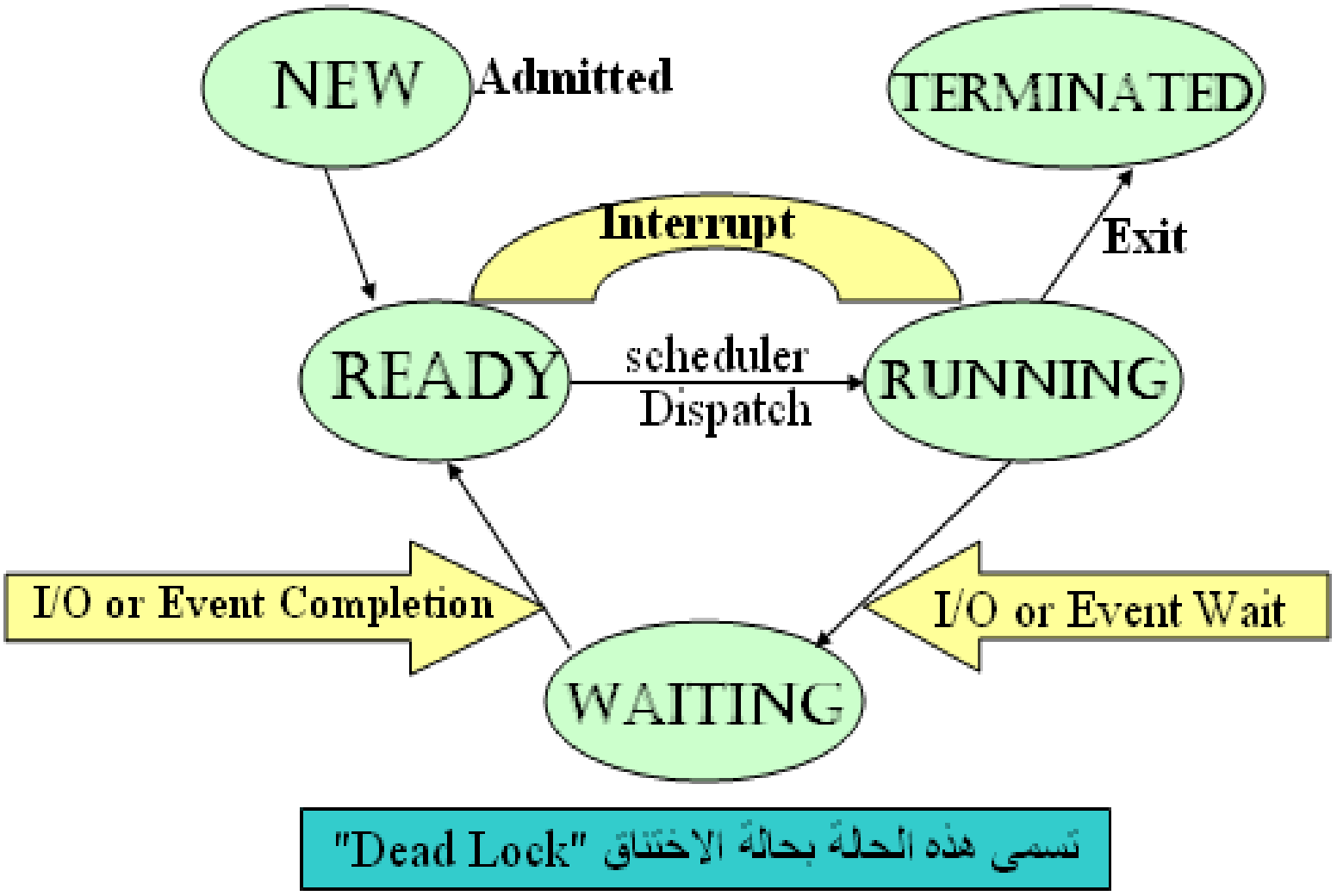
١ -تكبير استخدام الـC.P.U تحت قيد أو شرط إن اكبر زمن استجابة هو ثانية واحدة.

٢ -تكبير زمن الاستجابة قبل زمن الدوران والتي تكون مناسبة مع الزمن الكلي للتنفيذ.

الاختناق
DEAD LOCK

في محيط البرمجة المتعددة هنالك عدد من العمليات (Processes) تتنافس على العدد المحدد من المصادر .وعليه تقوم هذه البروسس بطلب المصادر فإذا لم تتوفر هذه المصادر في لحظة الطلب فان هذه البروسس تدخل حالة الانتظار .

كما إن المصادر المطلوبة مشغولة من قبل عمليات كانت منتظرة لها وكما بالشكل التالي:



تسمى هذه الحالة بحالة الاختناق "Dead Lock"

وعليه و حسب النموذج الاعتيادي للعملية تستطيع العملية استخدام المصادر وحسب التسلسل التالي:

١- الطلب Request

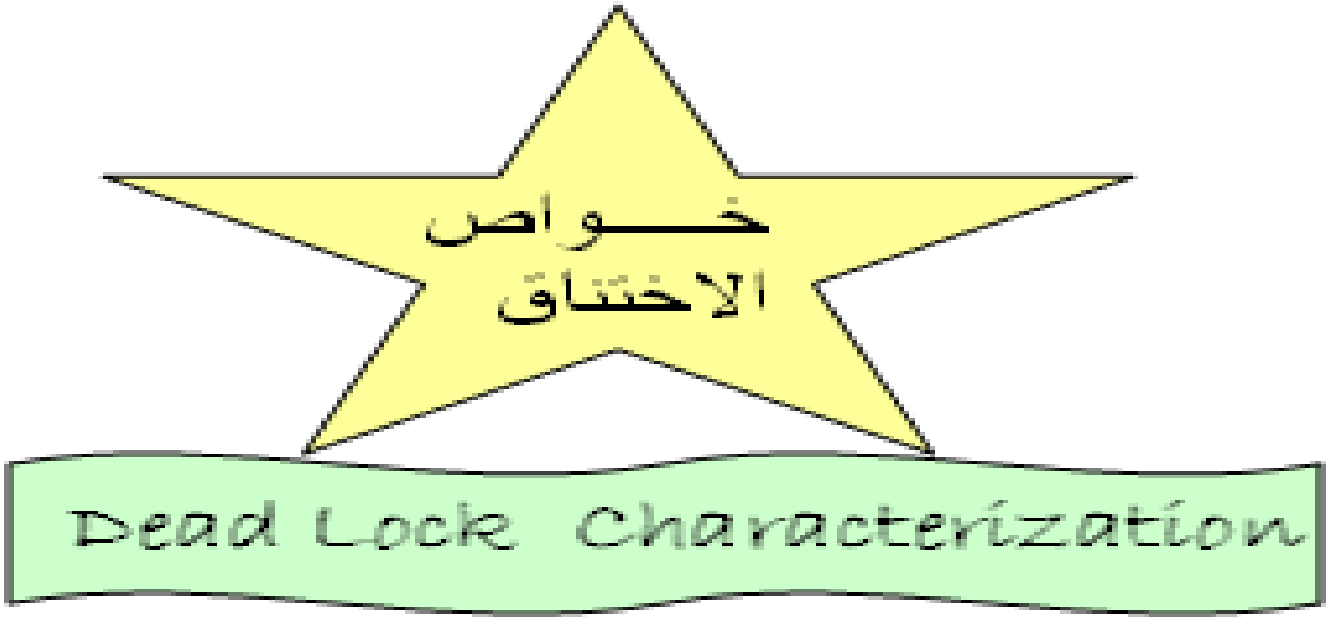
فعند حدوث طلب لمصدر معين و في حالة عدم استطاعة توفير المصدر أو الاستجابة لهذا الطلب حالا كون إن هذا المصدر مستخدم من قبل عملية أخرى فيؤجل الطلب "Wait" أي تدخل العملية حالة الانتظار "Wait State" حتى يتم تنفيذ الطلب.

٢- الاستخدام Use

تستطيع العملية العمل واستخدام المصادر المطلوبة "كالطابعات "Scanner، .

٣- إطلاق السراح Release

وعليه تكون "Release Request" هي نداءات نظام "System Calls".تطلق العملية المصدر بعد الانتهاء منه.



الشروط الضرورية
NECESSARY CONDITION

١ـ الاستثناء المشترك أو المتبادل:

Mutual Exclusion

وذلك يعني أن هناك عملية واحدة في كل مرة يستخدم المصدر وعليه أذا قامت عملية أخرى بطلب المصدر فان العملية تؤخر حتى يتوفر المصدر.

٢ - اخذ و الانتظار:

Hold and Wait

تقوم العملية بأخذ على الأقل مصدر واحد وتنتظر لأخذ مصادر أخرى وان تكون محجوزة من قبل عملية أخرى.

٣ -عدم الاحتجاز أو الصلاحية:

No Preemption

لا تحتجز العملية المصادر أي إن المصادر تطلق بشكل اختياري من قبل العمليات التي تحتجزها و لا تعطي صلاحية للمصادر في اختيار العملية.

٤ -الانتظار الدوار:

Circular Wait

ليكن لدينا مجموعة من العمليات(Pn....,p2,p1,p0) فإذا كانت P0تنتظر مصدر محجوز من قبل ***P1*** و ***P1*** تنتظر مصدر محجوز من ***P2*** وهكذا أي أن تصل إلى ***Pn*** تنتظر مصدر محجوز من ***P0*** فبذلك يكون هذا الانتظار دائري.

ملاحظة *Note* :

المبرمج الذي يقوم بتطوير تطبيقاته "*Application*" ذات المسارات المتعددة "Multi threaded Application" يجب إن يوجه انتباهه إلى مشكلة الاختناق وعليه فان التطبيقات أو البرامج ذات المسارات المتعددة هي أفضل حل لمشكلة الاختناق وذلك بسبب أن المسارات المتعدد تستطيع أن تتنافس على مصادر مشتركة "Shared Resources".

Request() , Release() , Device , Open() , and Close() , File() , Allocate() , and Free() , Memory are System Calls .

التخلص "الشفاء" من الاختناق
Recovery From Dead Lock

عندما يتم اكتشاف حالة الاختناق من خلال استخدام خوارزمية تحديد أو اكتشاف الاختناق هناك عدة حالات يمكن الاعتماد عليها لمنع حدوث الاختناق و كما يلي:

١ ـ إبلاغ المشغل(Operator)عن حدوث حالة الاختناق ونترك الأمر للمشغل لحل هذه المشكلة.

٢ ـ ترك النظام التشغيلي يقوم بحل هذه لمشكلة بشكل أوتوماتيكي وعليه يكون هناك اختيارات لقطع حالة الاختناق الدوري.

* إجهاض أو إلغاء Process أو أكثر لقطع حالة الاختناق الدوري.

** حجز بعض المصادر من بروسس أو أكثر من البروسس التي سببت الاختناق.

تجنب الاختناق
Dead Lock Avoidance

لتجنب الاختناق هناك خوارزميات يمكن من خلالهما تجنب الاختناق:

- الحالة الآمنة Safe State

تكون الحالة مؤمنة إذا استطاع النظام تحديد المصادر لكل بروسس وبطريقة الترتيب وتستطيع أن تبقى مانعة لوقوع الاختناق أي تأمين امن العمليات "Process"و عليه العمليات المتسلسلة (P1 , P2, ...,Pn) تكون بحالة أمان من الاختناق إذا استطاع النظام تأمين المصادر المطلوبة لكل عملية أي بإمكان البروسس إن تحقق طلبها من خلال المصادر المتوفرة أضف إن المصادر التي تكون محجوزة من قبل البروسس Pj بحيث إن المصادر هذه تكون غير متوفرة حاليا للبروسس Pi وعليه تستطيع Pi انتظار المصادر هذه حتى تنتهي منها Pj ولكن بشرط أن j>I وبعدها تستخدمها Pi.

مثال:

لنفرض أن هناك نظام لديه 12 CDs و 3 Processes (P0,P1,P2) ولكن P0:تطلب "تحتاج" 10 CDs.

P1:تحتاج 4 CDs

P2: تحتاج 9 CDs

الحل:

نفرض عند الزمن T0 كانت كل بروسس تحتاج فعلا للأقراص وحسب التالي: P0 أخذت 5 CDs و P1 أخذت 2 CDs و P2 أخذت 9CDs
إذا هناك 3 CDs غير مستخدمة...

	Maximum Needs	Current Needs
P0	10	5
P1	4	2
P2	9	2

وعليه يكون النظام بحالة امن عند الزمن T0 أي إن Process (P0, P1 P2) حققت شرط الأمان.

- مخطط تحديد المصدر

Resource – Allocation Graph

يمكن وصف الاختناق"Dead Lock" وبشكل دقيق في مصطلحات الرسم أو المخطط والذي يطلق عليه بنظام تحديد المصدر ألرسومي (A System Resource- Allocation Graph) وهذا الرسم أو المخطط يتكون من مجموعة من المتجهات "V" يمكن أن تجزأ إلى نوعين مختلفين من العقد "Nodes"وهي كالأتي:

1) P={P1, P2,P3,……,Pn}

وتظم هذه المجموعة العمليات الفعالة (Active Processes) في النظام.

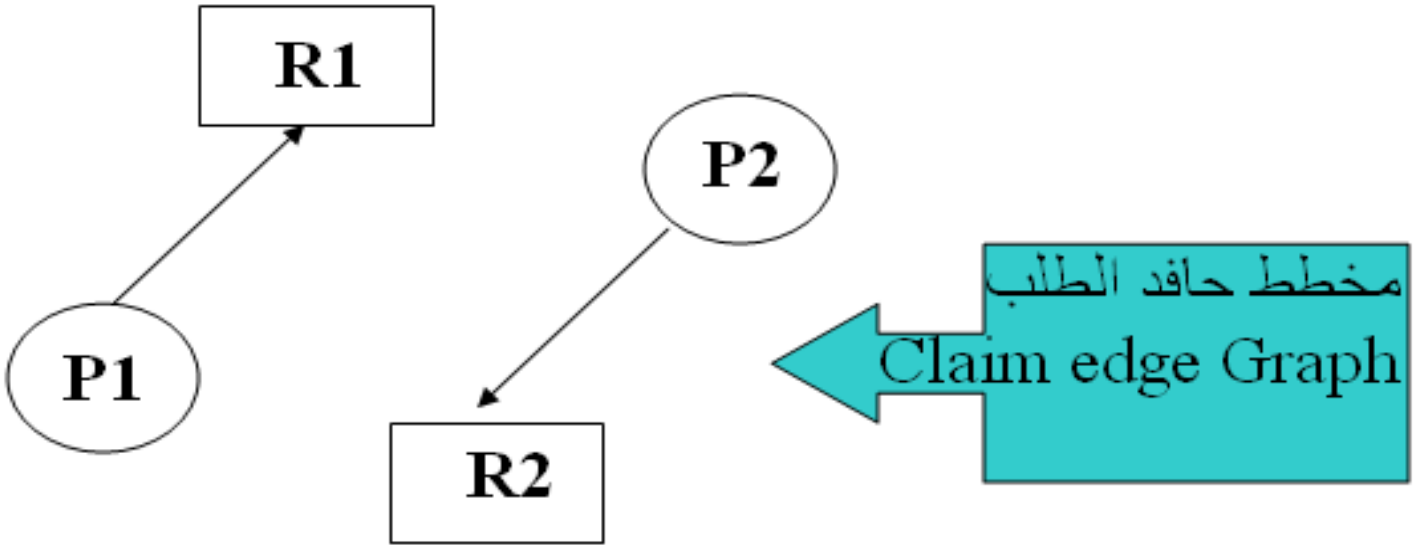
2) R={R1, R2, R3,…..,Rm}

وتظم مجموعة المصادر (Resource) في النظام وعليه يمكن تمثيل العلاقة الاتجاهية بين العملية والمصدر كالأتي:

إذا كان لدينا Pi من العمليات و Rj من المصادر فتكون العلاقة الاتجاهية كالآتي :

Pi ⟶ Rj

وهذه العلاقة تظهر طلب العملية Pi للمصدر Rj وهي تكون بذلك في حالة انتظار لهذا المصدر. والشكل التالي يوضح ذلك مفهوم الحافة المطلوبة Claimed أي أنProcess ⟵ (Pi) يطلب المصدر R1 و البروسس يطلب المصدر R2.



خوارزمية الأمان
Safety Algorithm

نستطيع ألان إظهار الخوارزمية لتحديد حالة النظام فيما إذا النظام بحالة أمنة أم لا و لتوضيح هذه الخوارزمية نتبع الخطوات الآتية:

1- Let Work and Finish be vectors of length M, N, respectively.

Initialize wor =Available and finish[I] = false

For I= 0,1,……,n-1

2-Find an I such that both a. finish[i]==false

b. need I<=work

If no such I exists , go step 4

3-Work =wor + allocation :

Finish [i] = true

Go to step 2

4-If finish [i]== true for all I, then the system is in

SAFE SATAE.

إكتشاف الإختناق
Dead Lock Detection

إذا كان النظام لا يستعمل حالة أو خوارزميات منع الاختناق DEAD LOCK PREVENTION هو تجنب الاختناق " Dead Lock Avoidance" أذن سوف تكون حالة الاختناق واقعية لا محالة . وعليه يجب على النظام توفير ما يلي:

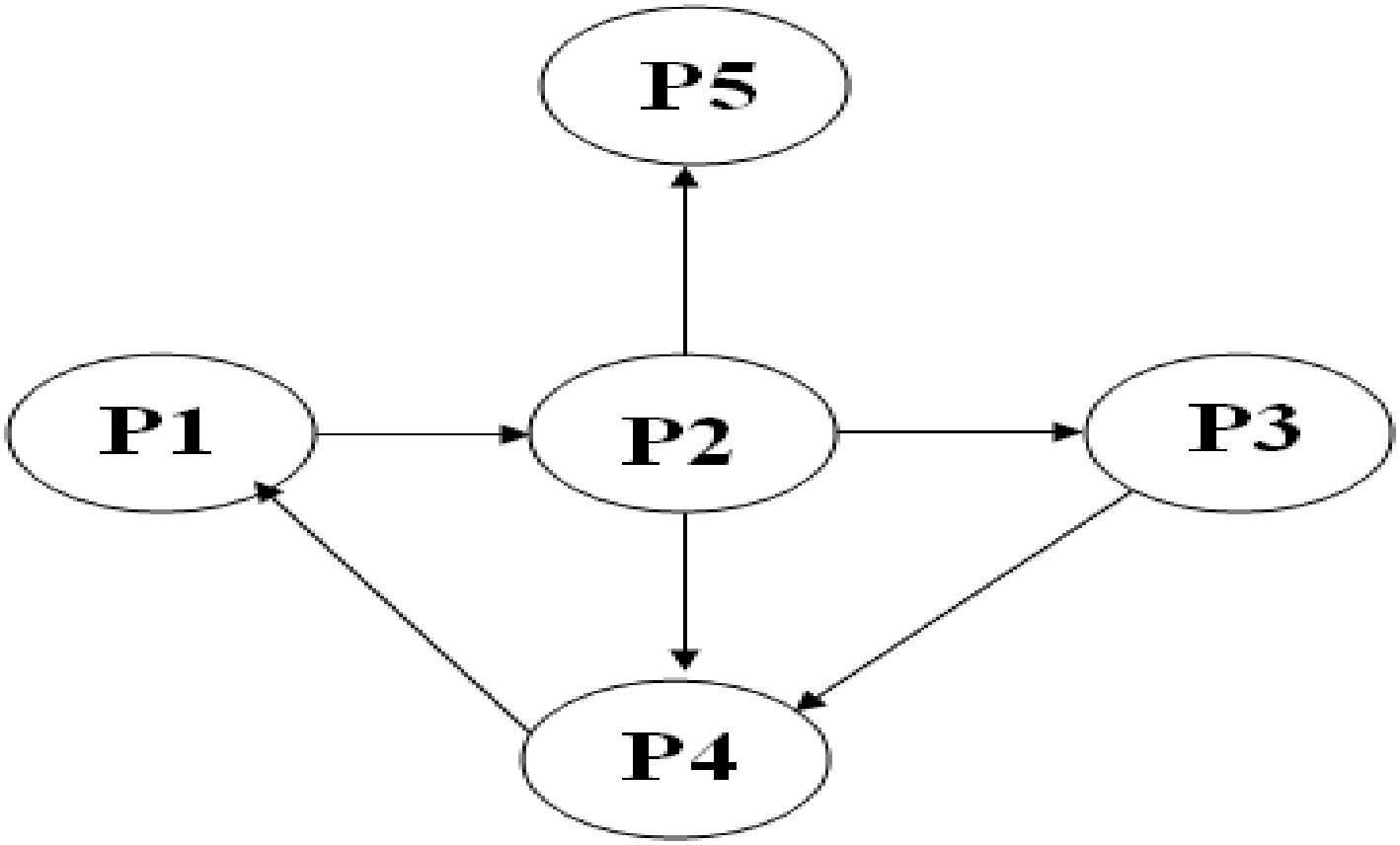
- خوارزمية فحص حالة النظام لتحديد فيما إذا أن حالة الاختناق حدثت.
- خوارزمية منع حدوث الاختناق .

وعليه هناك خوارزميتان تستخدمان لتجنب الاختناق

1- الطلب و المشاهدة المقررة لأي نوع من المصادر .

Single Instance of each resource type

تضمن هذه الخوارزمية على مبدأ المشاهدة المقررة لكل مصدر وعليه يمكن تعريف خوارزمية اكتشاف الاختناق والتي تستخدم التباين في مخطط **Waifor** وعليه يمكن الحصول على مخطط من خلال إزالة العقد للمصادر والشكل التالي يوصفه **Waifor**:



Wait – For Graph

مخطط انتظار للمصدر

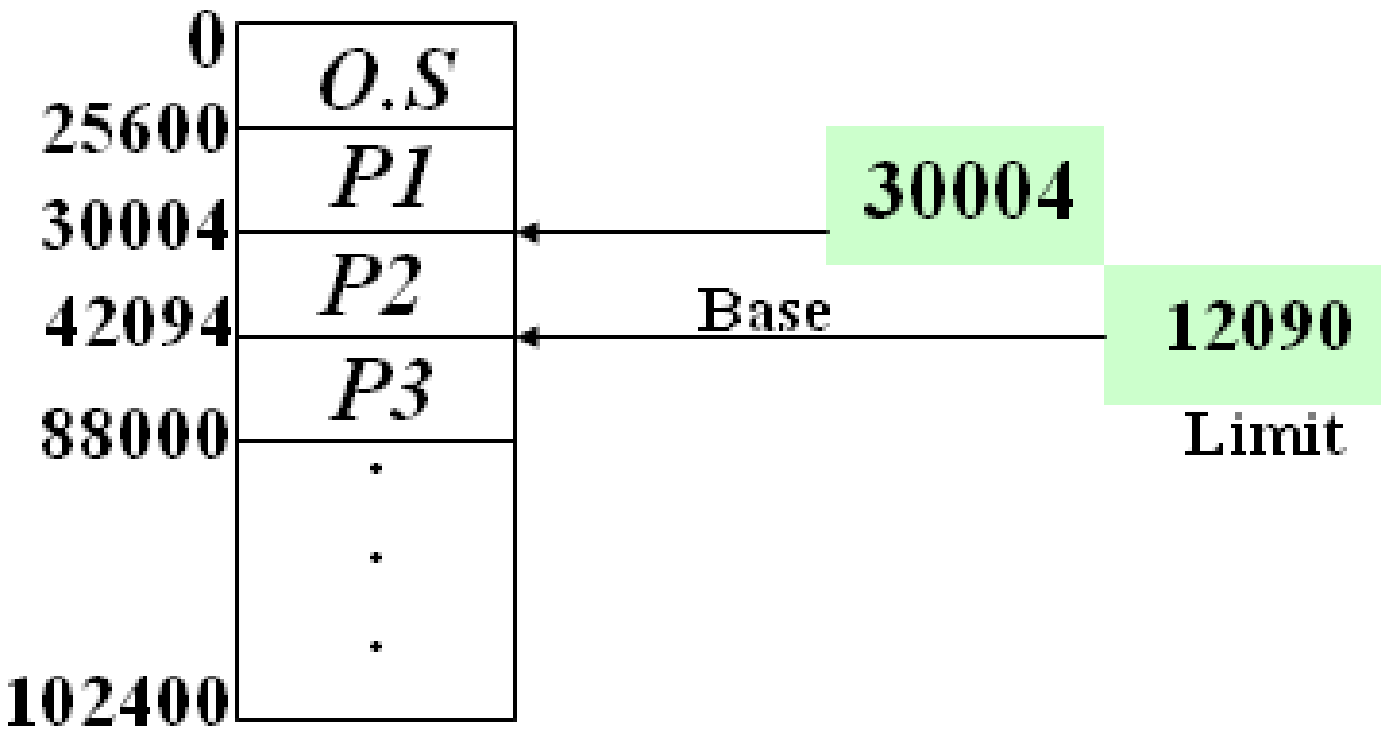
Memory Management

The main purpose of a computer system is to execute program . these programs together with the data they access , must be in main memory "at least partially " during execution .

- الذاكرة الرئيسية "Main Memory"
- الأساس المادي "basic Hard ware"

إن الذاكرة الرئيسية و السجلات "Register" التي بنيت داخل البروسس وكما عرفناها في موضوع "Fetch Cycle" هي وحدها فقط المخازن التي يدخلها C.P.U بشكل مباشر ولتكييف السرعة المتباينة بالـC.P.U والذاكرة الرئيسية Ram وضعت ذاكرة تسمى بـ Catch لهذا الغرض.

فتتكون الذاكرة الرئيسية من مصفوفة ذات بعد واحد تسمى "Array" من الكلمات Words أو البايتات Bytes وعليه يجب أن تحصل كل بروسس في الذاكرة على مساحة منفصلة و لتحقيق ذلك يجب أن تحدد مدى العناوين التي تتقلها البروسس لأن الذاكرة تحدد من خلال العناوين وكل عنوان له مساحة محددة تسمى Word ويكون بداية العنوان أي العنوان الأول هو الـ base reg. وعدد العناوين المشغولة من قبل البروسس يسمى Limit reg. وتعني Limit : Limit reg. و Base Register: Base reg. Register وكما موضح بالشكل التالي:



0 Base and A limit register define a logical address space .

نلاحظ في الرسم أن الـ Limit reg. هو المدى أو الجذر الذي يشغله البروسس P2 أي بدأت P2 من Address : 30004 وشغلت مساحة مقدارها 12090 والتي تكون بداية للبروسس التالية P3 :

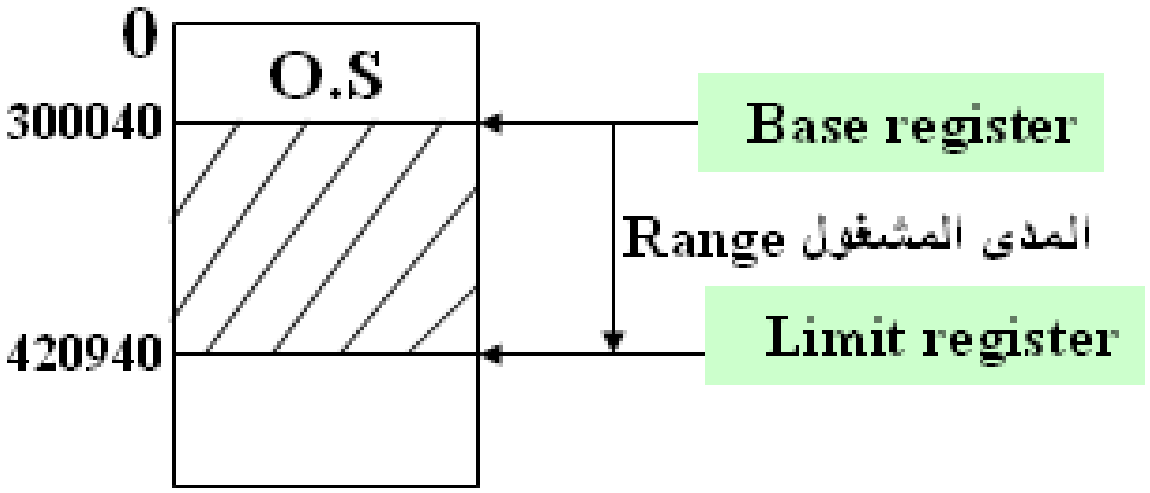
Base reg.P3=Base reg.P2 + Limit reh.P2

= 30004 + 12090 =42094

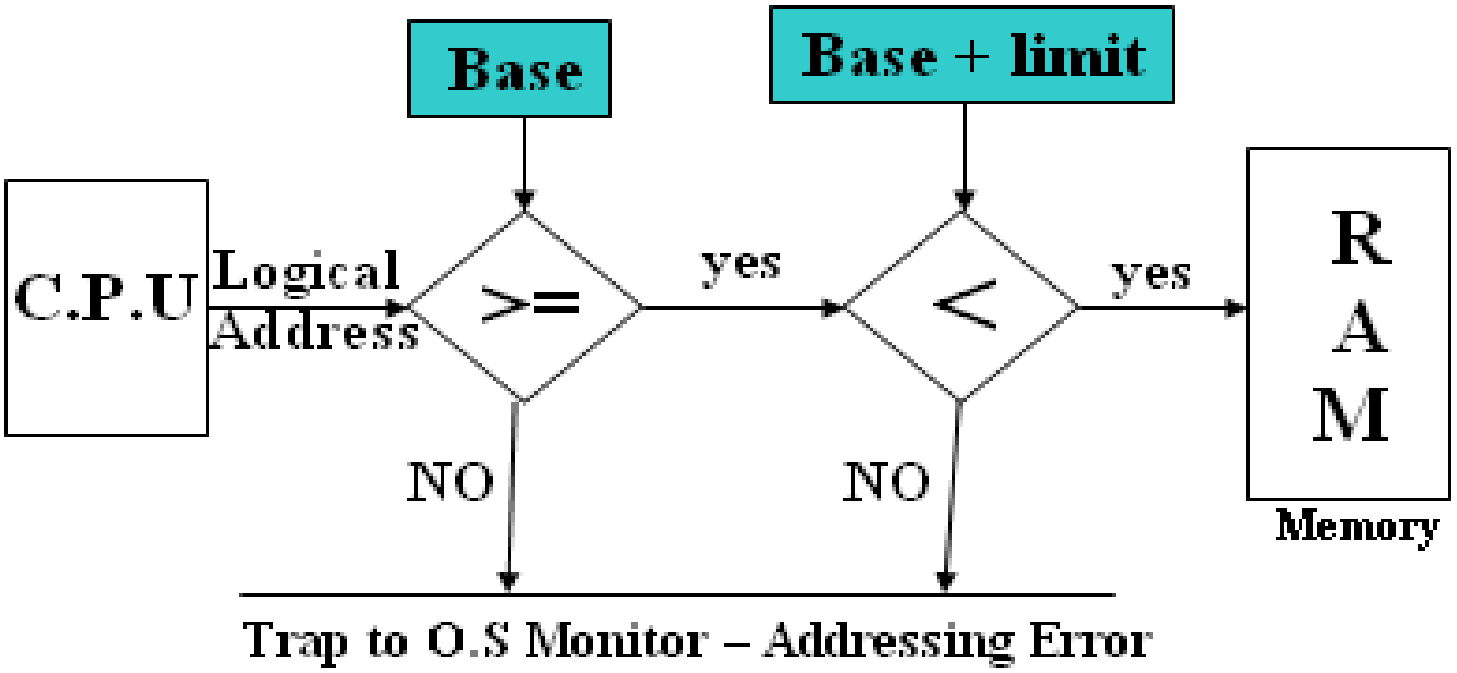
ولحماية أي بروسس في الذاكرة نستخدم base reg. و limit reg. لكي تحمي أي بروسس من الاختراق ، وعليه يكون base reg. يظم عنوان الذاكرة الفيزيائية الصغير المسموح به بينما يحدد Limit reg. حجم المدى المحجوز لهذه البروسس.

مثـــال :

If the base register had 300040 and limit register is 120900 , then the program can legally access all address from 300040 through 420940 "inclusive" meaning the program access the following address.



عليه فان الـ base reg. و limit يمكن أن يحمل من قبل الـO.S باستخدامات تعليمات امتيازيه خاصة في الـ Kernel mode و هو المكان الذي يمكن لنظام التشغيل أن ينفذ فيه .
وعليه يمكن لـO.S أن يغير الاقيام المسجلات ولكن يمنع المستخدم للبرنامج من ذلك وهذا امتياز للـO.S من السماح له أن يحمل load User's Program في User Memory وكذلك خزنها خارجه Dumpout في حالة حدث خطأ ليتمكن المستخدم من استدعاء البرنامج و المتغير منه.
الشكل التالي يوضح حماية العناوين الفيزيائية باستخدام base register and limit reg. وعليه فأن C.P.U الذي يكون العناوين المنطقية ثم بعد ذلك تحول إلى عناوين فيزيائية باستخدام صيغ خاصة. وكما موضح بالشكل التالي:



Hard Ware Address Protection With Base and Limit Register

Logical Versus Physical Address spaces

تتولد العناوين المنطقية في الـ C.P.U وعندما يرى العنصر ان من قبل وحدة الذاكرة Memory Unit ويحمل إلى الذاكرة بعدها تسمى "بالعنوان الفيزيائي " أي تم طبعه على الذاكرة كما في حال البريس عندما يتم تحويل الشكل المرسوم"المنطقي" إلى شكل واقعي "فيزيائي" كما يحصل في النقش على الذهب و الفضة و عليه فأن: Load Time & Compile Time .

Address-binding: هي طرق لتوليد العناوين المنطقية والفيزيائية المتناظرة تستخدم تسمية العنوان المنطقي Logical address أو الافتراضي Virtual Address وهما اسمان لشئ واحد، وعليه فان مجموعة العناوين المنطقية المتولدة من قبل البرنامج هي عبارة عن مساحة العناوين المنطقية والمجموعة لكل العناوين الفيزيائية المقابلة لتلك العناوين المنطقية هي مساحة العناوين الفيزيائية بمعنى :

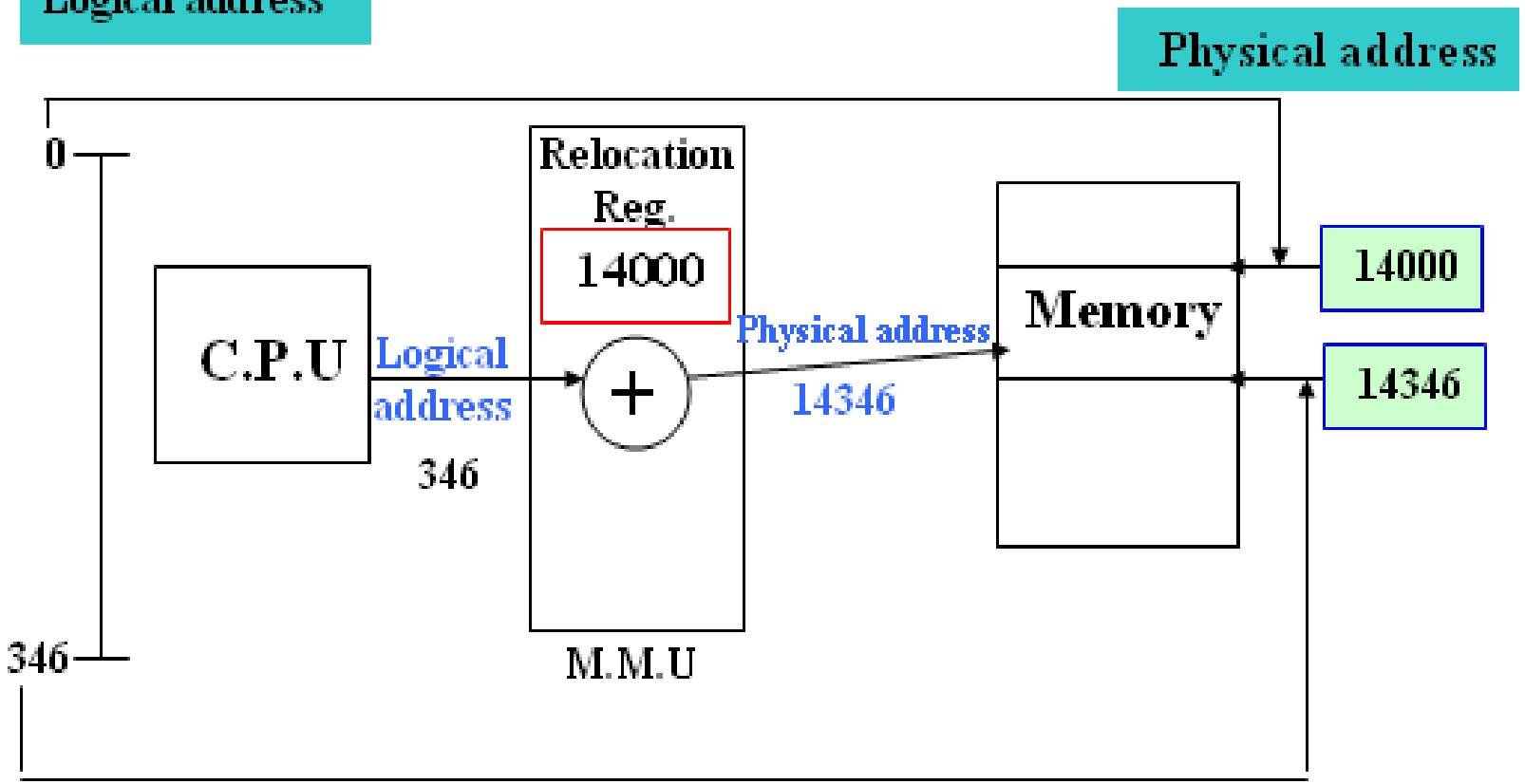
Logical address space ⟷ Logical address space

وعليه فان الـ Run – Time "زمن التنفيذ" تحويل "Mapping" العناوين المنطقية إلى عناوين فيزيائية و تنفذ من خلال جهاز مادي يسمى بـ "M.M.U" أي:

Memory- Management – Unit

وعليه فان زمن التنفيذ Exec.-Time و Addr.-binding تتكون من عناوين فيزيائية ومنطقية مختلفة و يمكن توضيح عملية التحويل Mapping باستخدام M.M.U والتي تعمل على توليد base reg. والذي نطلق عليه اسم Relocation reg. وعليه فان القيمة التي يضمها "base reg." Relocation reg. تضاف إلى أي عنوان من قبل "User Process" عند الزمن الذي أرسل العنوان فيه إلى الذاكرة .

وكما في الشكل التالي #:



Dynamic Relocation Using Relocation Register

وكما في المثال التالي :

If the base is at" 14000" then an Relocation attempt by the User to address location "0" is dynamic relocated to location "14000" an access to location "346" is mapped to location "14346".

أي بمعنى لو انك كتبت برنامج يشغل العنوان "صفر" في حساباتك إلى العنوان "436" وهذا هو العنوان المنطقي أي(0---><346) هذه العناوين المنطقية التي تولده من قبل البرنامج الذي كتبته بواسطة C.P.U فعند تحويله إلى الذاكرة فان وحدة أدارة الذاكرة سوف تحدد المكان الفارغ في الذاكرة أي الـbase reg وليكن "14000" وعليه فان عنوان "0 "أي برنامجك سيكون"14000" أي سوف يعمل له أعادة موقعيه"Relocation" ويحول Mapping إلى "14000" ودخول الموقع 346 سوف يحول إلى 14346.

وكما في الشكل أعلاه "#"وعليه فان المستفيد لا يرى حقيقة ما يحدث .

لكن البرنامج يكون مؤشر Pointer للموقع 346 ويخزنه و يعالجه في الذاكرة وعليه فان المستفيد يتعامل مع Logical address و إن Memory- Mapping ماديا تحول إلى Logical address إلى Physical address وعليه نحن ألان لدينا عنوانين مختلفين.

Logical address "in the range 0—> Max" and physical addresses

(R + 0 to R + Max) For abase Value R)

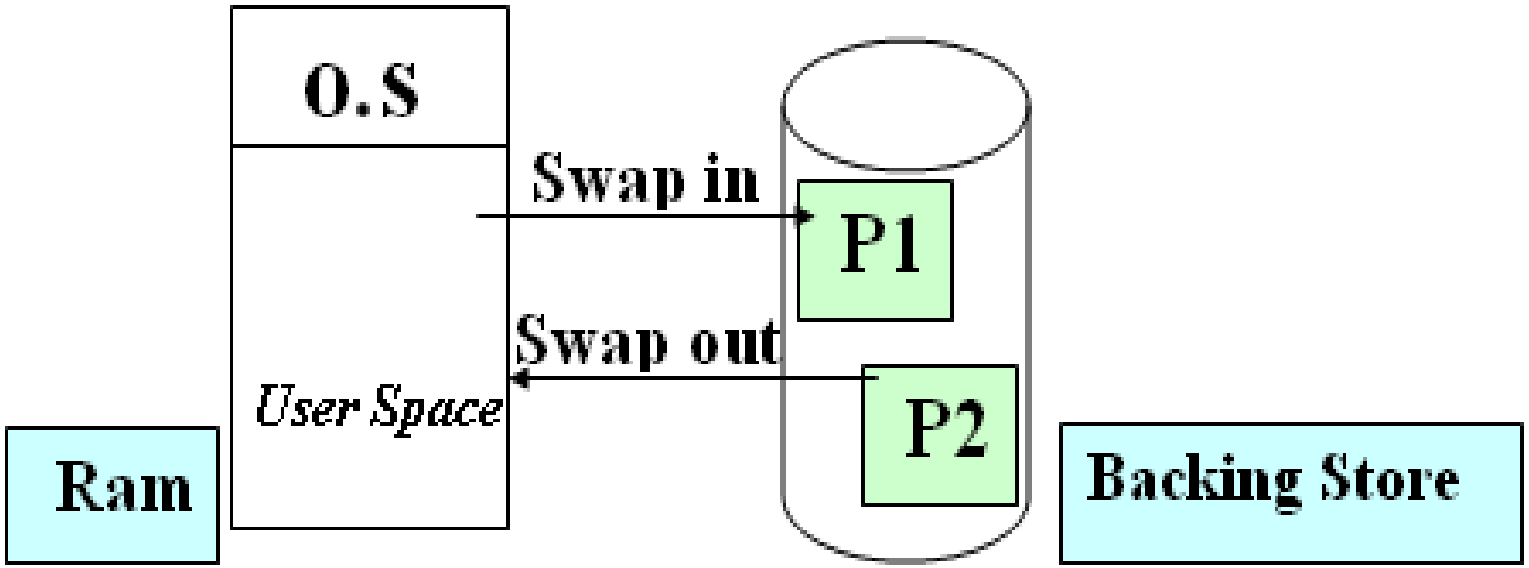
وعليه فان مستخدم البرنامج يعطي العنوان المنطقي في برنامجه وهذه العنوانين المنطقيين تحول "Mapping" إلى عنوانين فيزيائية قبل استخدامها.

التبادل في المواقع
Swapping

لكي نستطيع تنفيذ أي Process يجب أن تحمل هذه الـProcess إلى الذاكرة. وعليه فان Process تستطيع الانتقال "Swapped" وبشكل وقتي خارج الذاكرة الرئيسية إلى المخزن الخلفي "Backing Store" وبعدها تجلب إلى الذاكرة للاستمرار في تنفيذها.

Example:

Assume multiprogramming environment with a round robin C.P.U scheduling algorithm. When a quantum expires ,the memory manger will start to swap out the process that just finished and swap another process in to memory space that has been freed figure , to some other process in memory when each process finished it's quantum it will be swapped with another process.



Swapping of 2 processes using a disk as a backing store

Contiguous Memory Allocation
حصة الذاكرة المتصلة

يجب أن تتوائم الذاكرة الرئيسية مع كل من الـ O.S وبرامج المستخدم التي تكون تحت التنفيذ "User processes" وعليه نحن نحتاج إلى خصخصة أجزاء الذاكرة الرئيسية بطريقة أكثر ملائمة تقسم الذاكرة بشكل المعتاد إلى جزأين رئيسين هما :

- ✓ جزء مخصص للـ User Processes .
- ✓ جزء مخصص للـ O.S .

نستطيع وضع O.S في أي جزء ولكن اختيار موقع O.S في الجزء السفلي هو بسبب Interrupt Vector "متجه القطع" وعليه سوف يكون جزء الذاكرة المخصص للبرامج المستخدمين تكون متواصلة وعليه تكون أي Process في مقطع في الذاكرة المتواصلة أي أن الذاكرة تكون متواصلة و غير متواصلة.

خصصة الذاكرة
Memory Allocation

أن احد أهم الطرق خصخصة الذاكرة هي تجزئتها إلى أجزاء ثابتة Fixed- Size تسمى Partitions "أجزاء" وكل جزء ربما يظم بروسس واحدة. وعليه فان أي جزء من الذاكرة يكون غير مشغول Free سوف يتم اختيار Ready-queue-process وتحميلها في هذا الجزء الحر ،و لاختيار الأجزاء الغير مشغولة بالذاكرة هناك عدد من الاستراتيجيات المستخدمة لاختيار الأجزاء وهي :

١ - تخصيص أول جزء من الذاكرة يكفي لهذه البروسس "First Fit" .

٢ - تخصيص اصغر جزء يكفي لهذه البروسس "Best Fit".

٣ - تخصيص اكبر جزء من الذاكرة لهذه البروسس وان كان حجم البروسس اقل "Worst Fit".

وعليه أفضل طريقة أو إستراتيجية هي "First Fit, Best Fit".

التكسرات في الذاكرة
Fragmentation

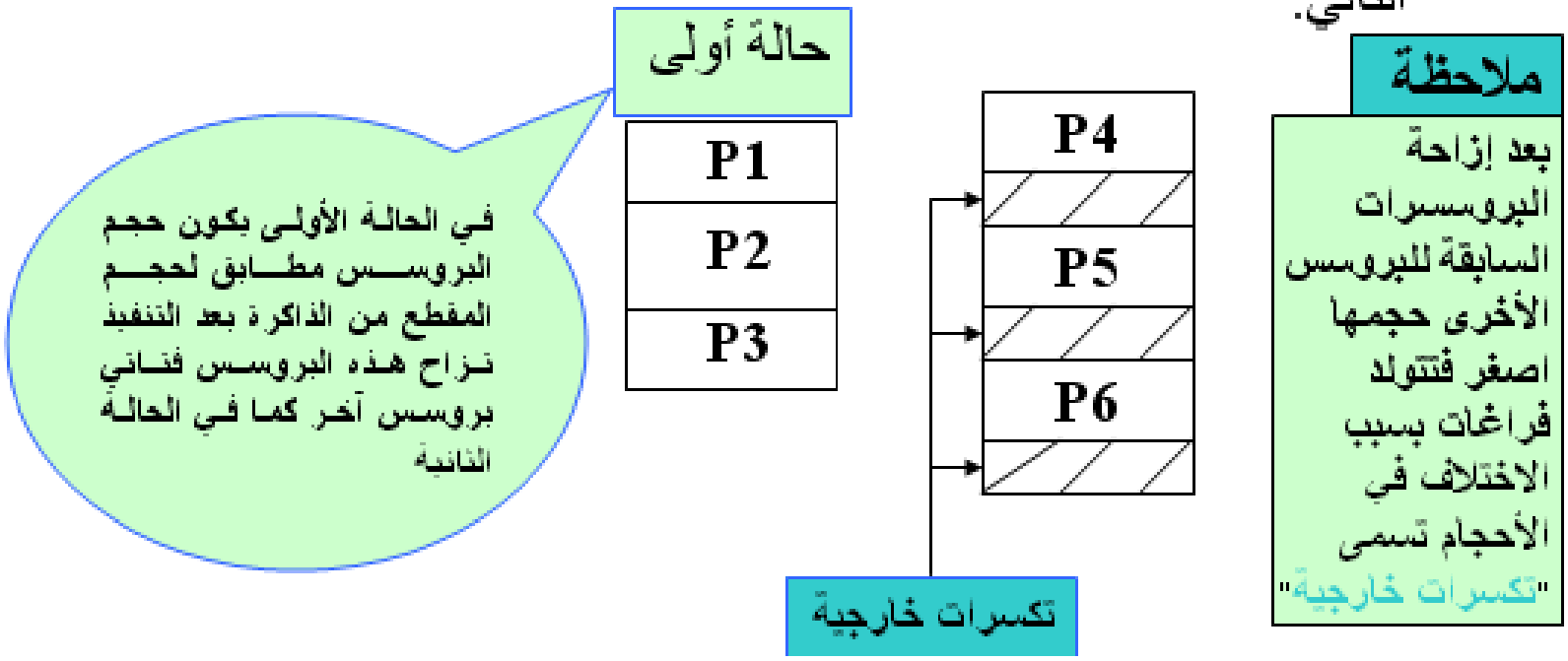
تعاني استراتيجيات تخصيص الذاكرة من التكسرات الخارجية "External Fragmentation" وعليه فان البروسس عندما تحمل إلى الذاكرة وتزاح من الذاكرة. وعليه فان مساحة الذاكرة سوف تكسر إلى أجزاء صغيرة.

وعليه فان مجموعة من التكسرات" External Fragmentation " حجمها يكفي لبروسس معينة ولكنها ليست متواصلة أي قطعة واحدة أي لا يمكن أن تخزن هذه Process في الذاكرة لماذا؟

لأنها عبارة عاكس صغير منفصلة.

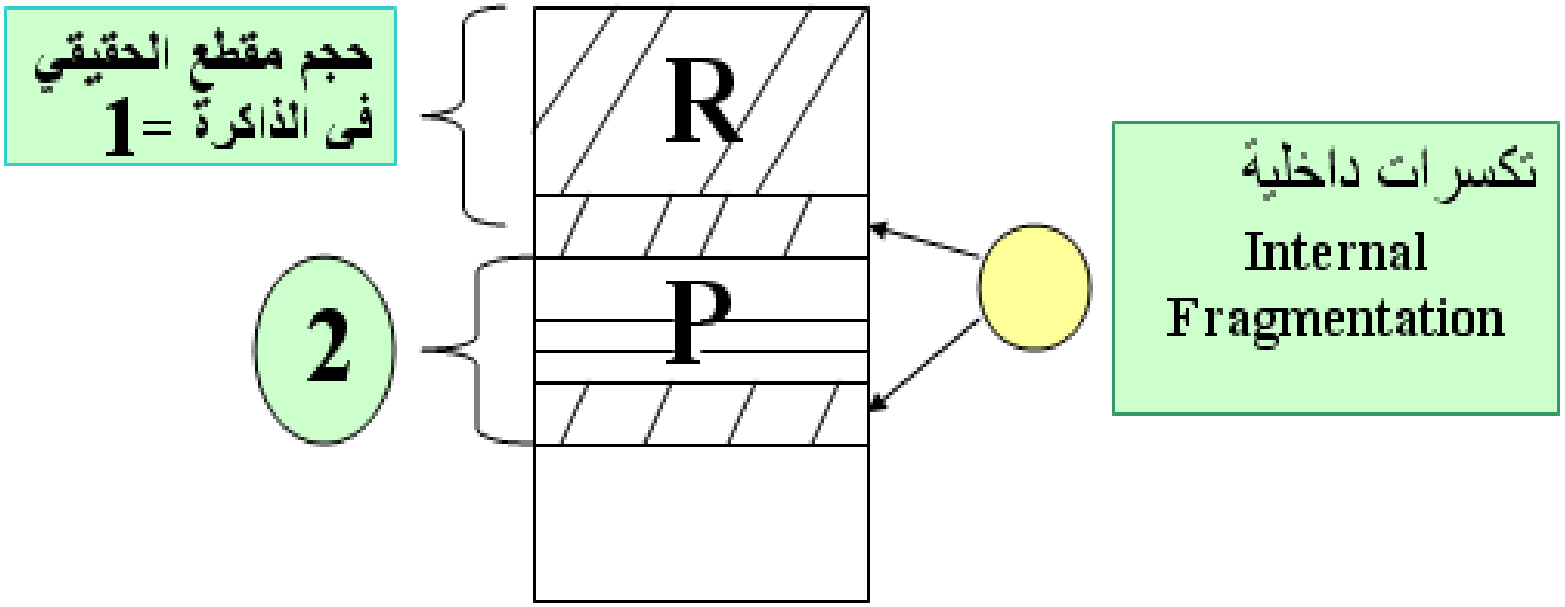
لكل في Worst Case نحن باستطاعتنا أن نحصل على مقطع من المساحة الحرة في الذاكرة "مهملة" من الذاكرة بين بروسسرين أو أكثر، وعليه فان الأسلوب الأفضل في تجنب مثل هذه المشكلة هو تقطيع الذاكرة إلى وحدات معتمدة على حجم المقطع.

وعليه فان المقطع المخصص لبروسس معينة ربما يكون حجمه اكبر من حجم البروسس التي خصص لها هذا المقطع وعليه فان الاختلاف بين هذين الحجمين حجم المقطع و حجم البروسس يسمى بالتكسر الداخلي Internal Fragmentation كما في الشكل التالي:



المساحات الحاصلة بسبب صغر حجم البروسس الأخرى مقارنة البروسس السابقة.
External Fragmentation

الشكل التالي يوضح Internal Fragmentation :



مقارنة حجم الذاكرة الحقيقي Fixed-Sized مع حجم البروسس اقل من حجم المقطع الفرق يسمى
Internal Fragmentation

و أخيرا

قاعـــــدة تحـــويل :

Logical Address ——> Physical Address

Physical Address=(frame no. * page size) + off set of logical address.