Computer Function

الوظيفة الاساسية التي يؤديها الكمبيوتر هي تنفيذ البرنامج يتكون من مجموعة تعليمات مخزنة في الذاكرة.

الـ processor المعالج يقوم بالعمل الفعلي الي هو تنفيذ (executing) التعليمات (instructions) المحددة التي في البرنامج.

سنقدم نظرة عامة على العناصر الاساسية لتنفيذ البرنامج.

تتكون معالجة التعليمات (instruction processing) في ابسط صورها من خطوتين: يقرأ المعالج (يجلب او fetches) التعليمات (instructions) من الذاكرة (memory) واحدة تلو الاخرى وينفذ كل التعليمات.

يتكون تنفيذ البرنامج من تكرار عملية الجلب التعليمات (instruction fetch) و وتنفيذ كل تعليمة (executes each instruction).

تسمى المعالجة المطلوبة لتعليمة الواحدة instruction cycle .

هاذي الصورة توضح دورة التعليمات (instruction cycle) ويشار الى الخطوتين بـ جلب التعليمات وتنفيذ التعليمات (fetch cycle and the execute cycle).

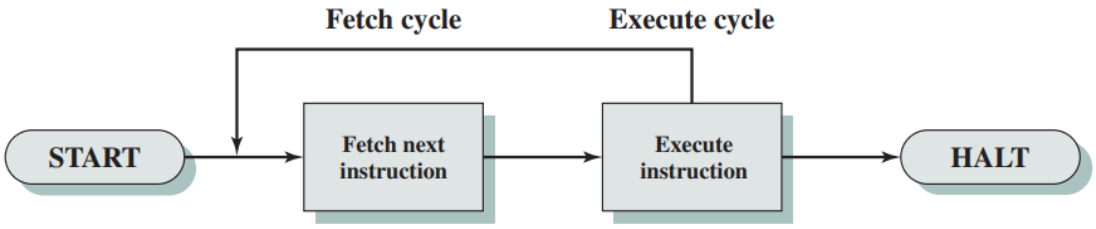


Figure 3.3 Basic Instruction Cycle

Instruction Fetch and Execute

في بداية كل دورة تعليمات يقوم المعالج بجلب التعليمات من الذاكرة.

في الـ typical processor يوجد سجل او Register يسمى program counter (PC) يحتفظ بعنوان التعليمات التي سيتم جلبها بعد ذالك.

على سبيل المثال كمبيوتر تشغل كل التعليمات 16-bit word من الذاكرة افترض الـ program counter PC تم تعينه على موقع الذاكرة 300 حيث يشير عنوان الموقع الى 16-Bit word .

سيقوم المعالج بجلب تعليمات من موقع 300 وفي دورة تعليمات التالية سيقوم بجلب المعالج التعليمات من موقع 301 بعدها 302 وبعدها 303 الخ وقد يتغير هذا التسلسل.

يتم تحميل التعليمات التي تم جلبها (fetched instruction) في سجل register في المعالج يعرف باسم instruction register (IR) .

الـ instruction تحتوي على وحدات البت التي تحدد الاجراء الذي يجب على المعالج ان يتخذه.

يقوم المعالج بتفسير التعليمات وتنفيذ الاجراء المطلوب. وبشكل عام تنقسم هذه الافعال الى اربع فئات :

Processor-memory : قد يتم نقل من المعالج الى الذاكرة والعكس .

Processor-I/O : قد يتم نقل البيانات من او الى جهاز طرفي (peripheral device) عن طريق النقل بين المعالج و وحدة I/O .

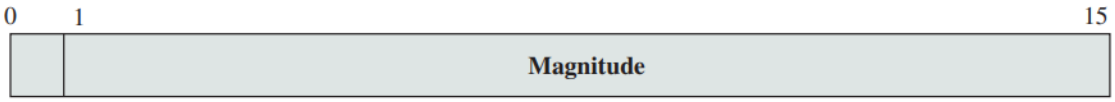
Data processing : قد يقوم المعالج باجراء بعض العمليات الحسابية او المنطقية على البيانات.

Control : قد تحدد التعليمات تغيير تسلسل التنفيذ. مثال يمكن للمعالج تعليمات في موقع 104 والتي تحدد ان التعليمات التالية ستكون في 502 وبتالي بدلا من الذهاب الى موقع 104 الى 105 سيقوم الذهاب الى 502.

قد يتضمن تنفيذ التعليمات مجموعة من هاذي المتطلبات.

فكر في مثال بسيط باستخدام اله افتراضية تتضمن خصائص المذكورة في الاعلى.

(a) Instruction format



(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

يحتوي المعالج processor على سجل Register واحد للبيانات يسمى accumulator (AC) .

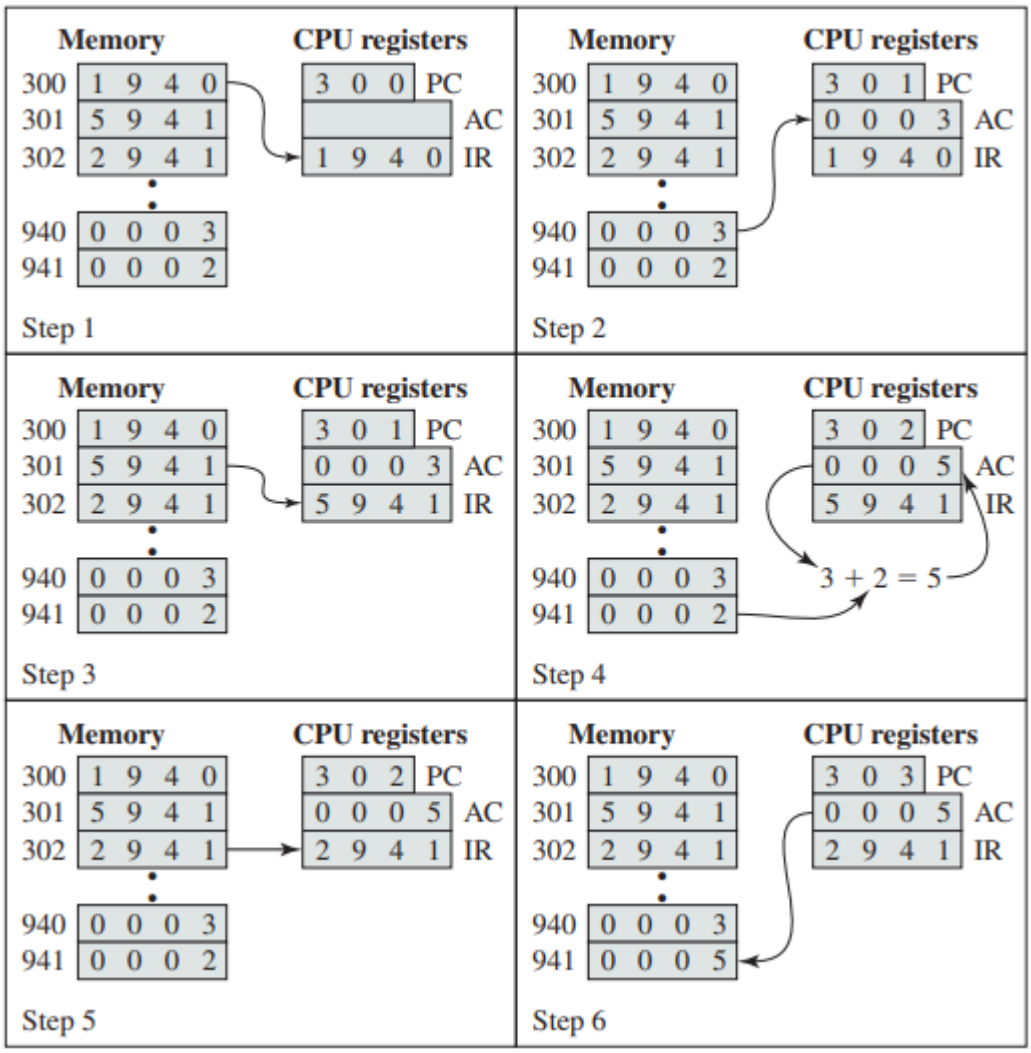
يبلغ طول الـ instructions والداتا 16-bits .

بتالي من الملائم تنظيم الذاكرة باستخدام 16-bits words .

يوفر الـ instruction format 4 بت لـ opcode .

بحيث يمكن ان يكون هناك ما يصل الى $2^4 = 16$ بمعنى 16 opcodes مختلفة .

ويمكن معالجة ما يصل الى $2^{12} = 4096$ kb .



الصورة هاذي توضح تنفيذ جزئي للبرنامج موضح الاجزاء ذات الصلة من الذاكرة والسجلات والمعالج.

البرنامج يضيف محتويات الذاكرة في عنوان 940 الى محتويات 941 ويخزن النتيجة الاخيرة .

هناك حاجة لتنفيذ ثلاثة تعليمات والتي يمكن وصفها بثلاث دورات جلب وثلاث دورات تنفيذ :

1 – من عنوان 300 يتم تحميل هذه التعليمات في سجل IR ويتم زيادة الـ PC لاحظ ان هاذي العملية تتضمن استخدام Memory address register و memory buffer register .

2 – سيتم تحميل قيمة 3 من عنوان 940 في سجل AC .

3 – من عنوان 301 سيتم وضع قيمته التي هي تعليمة في سجل IR وايضا سجل الـ PC يتم زيادته .

4 – تتم اضافة محتويات الـ AC وموقع 941 ويتم تخزينها في سجل AC .

5 – تتم جلب تعليمات التالية من موقع 302 ويقوم بوضعها في IR ويتم زيادة الـ PC .

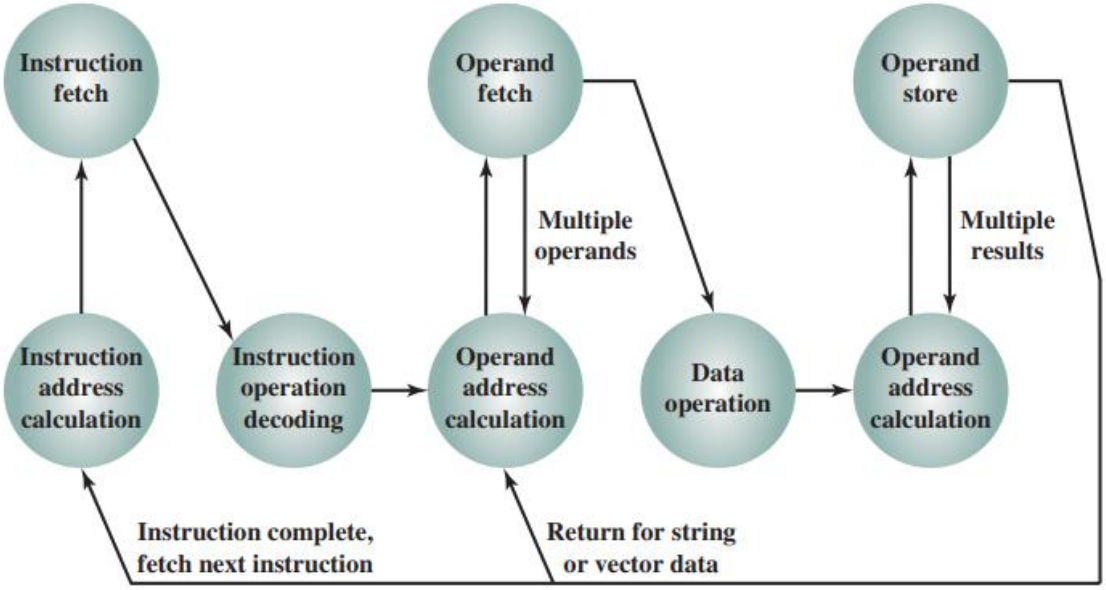
6 – يتم تخزين محتويات الـ AC في موقع 941.

في هذا المثال هناك حاجة الى ثلاث دورات تعليمات تتكون كل منها من دورة جلب ودورة تنفيذ لاضافة محتويات من 940 الى 941 .

بتالي فان دورة التنفيذ لـ instruction معينة قد تتضمن اكثر من مرجع واحد للذاكرة.

وايضا بدلا من مراجع الذاكرة (memory references) قد تحدد التعليمات I/O operation .

الصورة هاذي تقدم اكثر تفصيل عن عملية دورة التعليمات التي قدمناها في الاعلى بشكل مبسط.



بالنسبة لاي دورة تعليمات معينة قد تكون بعض الحالات فارغة وقدد تتم زيارة حالات اخرى اكثر من مرة.

وصف الصورة :

Instruction fetch (if)

اقرأ التعليمات من موقعها في ذاكرة الى المعالج .

Instruction operation decoding (iod)

تحليل التعليمات ولتحديد نوع العملية التي سيتم تنفيذها والمعامل (operand) او اكثر التي سيتم استخدامها.

Operand address calculation (oac)

اذا كانت العملية تتضمن اشارة الى Operand في الذاكرة او متاح عبر الـ I/O فحدد عنوان الـ Operand .

Operand fetch (of)

جلب المعامل من الذاكرة ا قراءته من الـ I/O .

Data operation (do)

قم بتنفيذ العملية (operation) الموضحة في التعليمات .

Operand store (os)

اكتب النتيجة في الذاكرة او I/O .

Interrupt

توفر كافة اجهزة الكمبيوتر تقريبا الية يمكن من خلالها للـ modules الاخرى مثل الـ (Memory & I/O) انها تقاطع (Interrupt) المعالجة العادية للمعالج.

جدول :

الان لا نحتاج الى فهم المقاطعات سنقوم بشرحها لاحقا لانها تحتاج الى مفهومات اكثر دقة في دورة التعليمات وتاثير المقاطعات على interconnection structure .

يتم توفير المقاطعات في المقام الاول لتحسين كفاءة المعالجة.

على سبيل المثال معظم الاجهزة الخارجية ابطأ بكثير من الـ Processor المعالج.

لنفترض ان المعالج ينقل البيانات الى الطابعة باستخدام مخطط دورة التعليمات هذا :

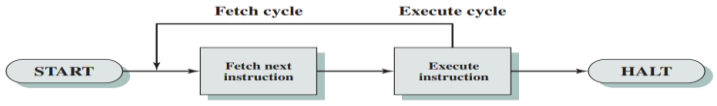
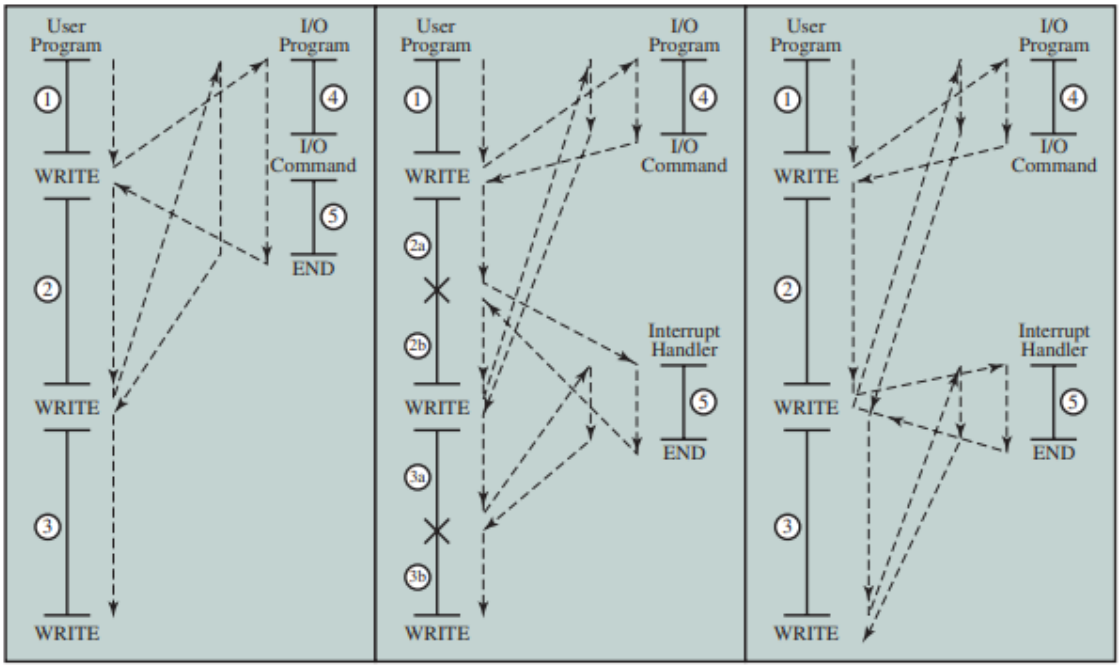


Figure 3.3 Basic Instruction Cycle

بعد كل write operation يجب ان يتوقف المعالج مؤقتا ويظل خاملا (idle) حتى الطابعة تسوي catches up (تنجح في الوصول) . قد تكون مدة هذا التوقف المؤقت في حدود عدة مئات او الاف الدورات التعليمات التي لا تتضمن الذاكرة.

الواضح انه هذا مسرف كبير للمعالج .



هاذي الصورة توضح مفهوم المقاطعة .

يقوم الـ user program باجراء سلسلة من مكالمات الـ WRITE المتداخلة مع المعالجة.

تشير اجزاء التعليمات 1 و 2 و 3 الى تسلسلات تعليمات لا تتضمن I/O .

يتكون الـ I/O Program من ثلاثة اقسام :

1 - سلسلة من التعليمات التي تحمل رقم 4 للتحضير لعملية الـ I/O الفعلية. قد يتضمن ذالك نسخ البيانات المراد اخراجها الى Buffer واعداد parameters لـ device command .
2 – امر I/O الفعلي (The actual I/O command) . بدون استخدام المقاطعات بمجرد اصدار هذا الامر يجب على البرنامج انتظار الـ I/O لأداء الوظيفة المطلوبة . قد ينتظر البرنامج عن طريق التحقق من عملية الـ I/O قد تمت ام لا.
3 – سلسلة من التعليمات التي في رقم 5 لأكمال العملية.

لانه عملية I/O قد تستغرق وقتا طويلا نسبيا حتى تكتمل فان برنامج الـ I/O معلق في الانتظار حتى اكتمال العملية.
وبتالي يتم ايقاف الـ user program عند نقطة WRITE Call لفترة زمنية طويلة.

interrupts and the instruction cycle

يمكن ان يشارك المعالج بتنفيذ تعليمات اخرى اثناء عملية I/O .

في الصورة التي في الاعلى في جزء (b) .

كما كان من قبل يصل البرنامج المستخدم الى نقطة التي يقوم بها باستدعاء النظام WRITE Call .

يتكون برنامج الـ I/O الذي يتم استدعاؤه في هذه الحالة فقط من preparation code و actual I/O command .

بعد تنفيذ هاذي التعليمات القليلة يعود التحكم الى برنامج المستخدم .

في الوقت نفسه يكون البرنامج الخارجي مشغولا بقبول البيانات من ذاكرة الكمبيوتر وطباعتها.

الـ I/O operation تتم بالتزامن مع تنفيذ التعليمات الموجودة في برنامج المستخدم.

عندما يصبح الجهاز الخارجي serviced اي عندما يكون جاهزا لقبول المزيد من البيانات من المعالج ترسل وحدة I/O الخاصة بهذا الجهاز الخارجي اشارة طلب مقاطعة الى المعالج.

يستجيب المعالج عن طريق تعليق تشغيل البرنامج الحالي والتفرع الى برنامج لخدمة i/o المعين هذا والمعروف باسم interrupt handler واستئناف التنفيذ الاصلي بعد ما يكون الـ device يصير serviced .

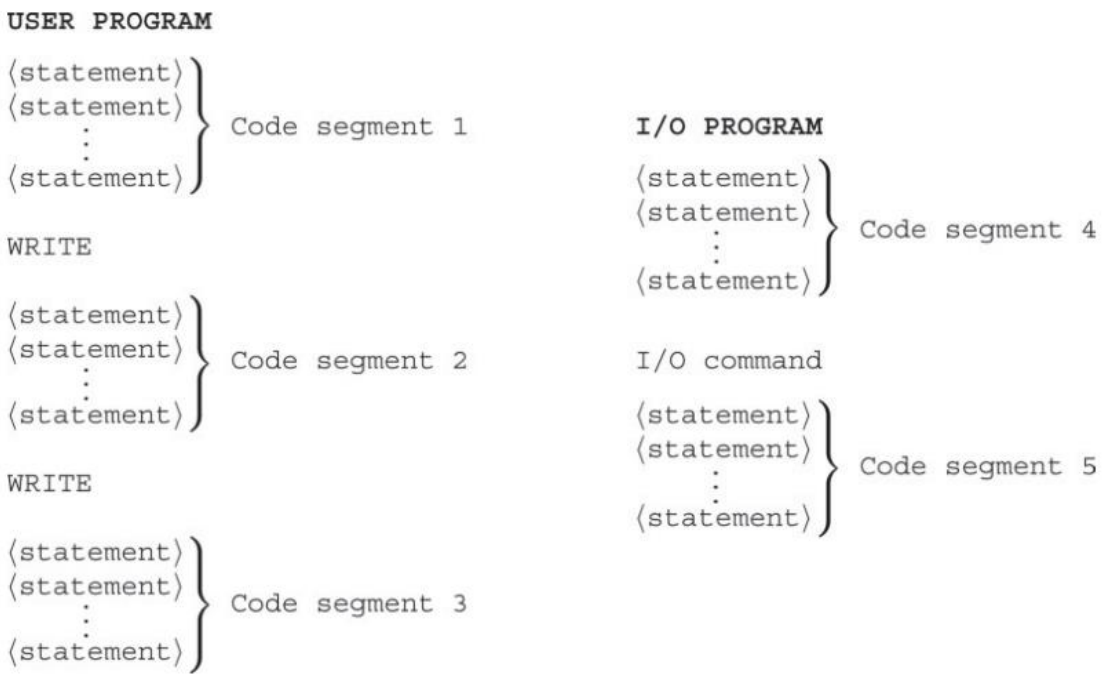
لنستمر في توضح الصورة التي في الاسفل لدينا برنامج مستخدم يحتوي على امرين WRITE .

يوجد segment of code في البداية ثم امر WRITE واحد من ثم segment of code بعدها امر ثاني WRITE ثم segment of code الاخير.

الـ WRITE command يستدعي I/O الذي يوفره نظام التشغيل.

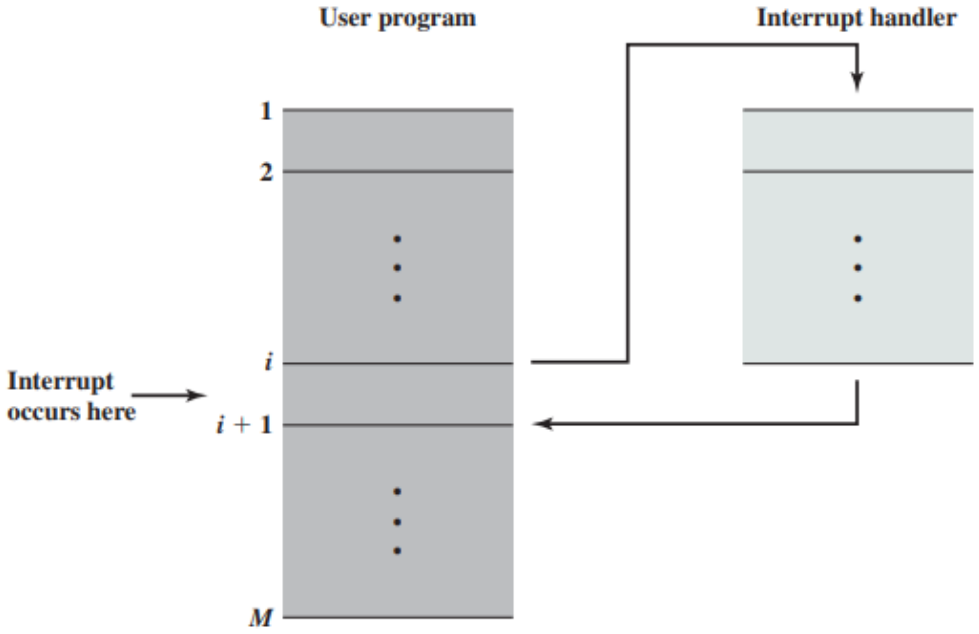
ونفس الشيء يتكون برنامج الـ I/O من مقطع التعليمات (segment of code) متبوعا بامر بـ I/O command ومتبوعا بجزء من segment of code .

الـ I/O command يستدعي hardware I/O operation .



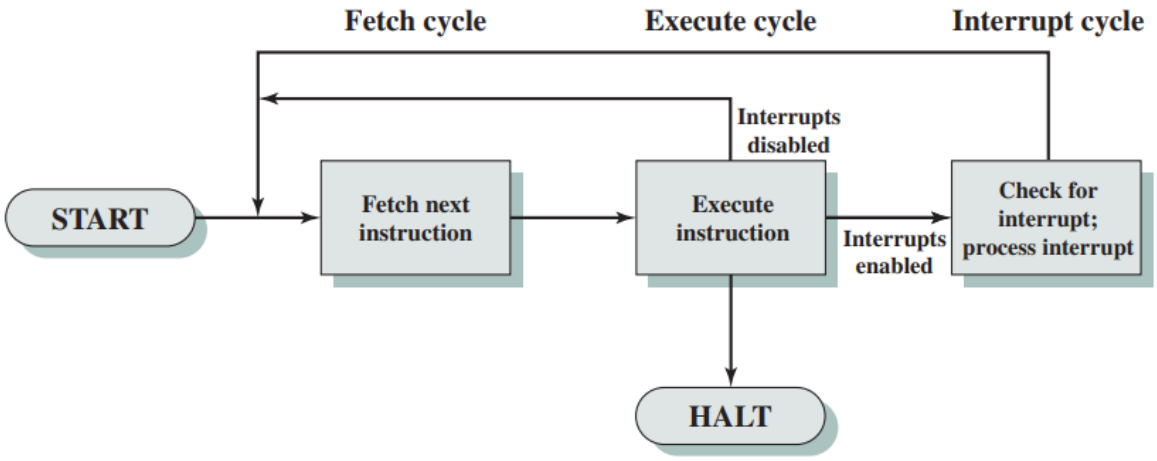
من وجهة نظر الـ user program المقاطعة (interrupt) هي فقط تسلسل طبيعي للتنفيذ.

عند اكتمال معالجة المقاطعة (interrupt) يتم استئناف التنفيذ مثل الصورة :



وبتالي ليس من الضروري ان يحتوي البرنامج على تعليمات برمجية يمكنها استيعاب المقاطعة المعالج ونظام التشغيل هم المسؤولان عن تعليق (suspending) البرنامج او استئنافه.

لاستيعات المقاطعة يتم اضافة دورة مقاطعة (interrupt cycle) الى دورة التعليمات (instruction cycle) مثال صورة :



في دورة المقاطعة يتحقق المعالج ما اذا كان يوجد هناك مقاطعة قد حدثت والتي تتم الاشارة اليها من خلال وجود اشارة مقاطعة (interrupt signal).

اذا كانت المقاطعة في حالة انتظار يقوم المعالج بما يلي :

1 – يقوم بتعليق التنفيذ البرنامج الحالي الذي يتم تنفيذة وحفظ الـ context الخاص به. وهذا يعني حفظ عنوان التعليمة التالية المراد تنفيذها (المحتويات الحالية لـ Program Counter) واي بيانات اخرى ذات صلة بالنشاط الحالي للمعالج .

2 – يقوم بتعيين سجل الـ PC لعنوان بداية الـ interrupt handler routine .

وينتقل الان المعالج الى دورة الجلب (fetch cycle) ويجلب اول تعليمة (instruction) في interrupt handler program الذي سيقوم بخدمة المقاطعة.

الـ interrupt handler program بشكل عام هو جزء من برنامج التشغيل.

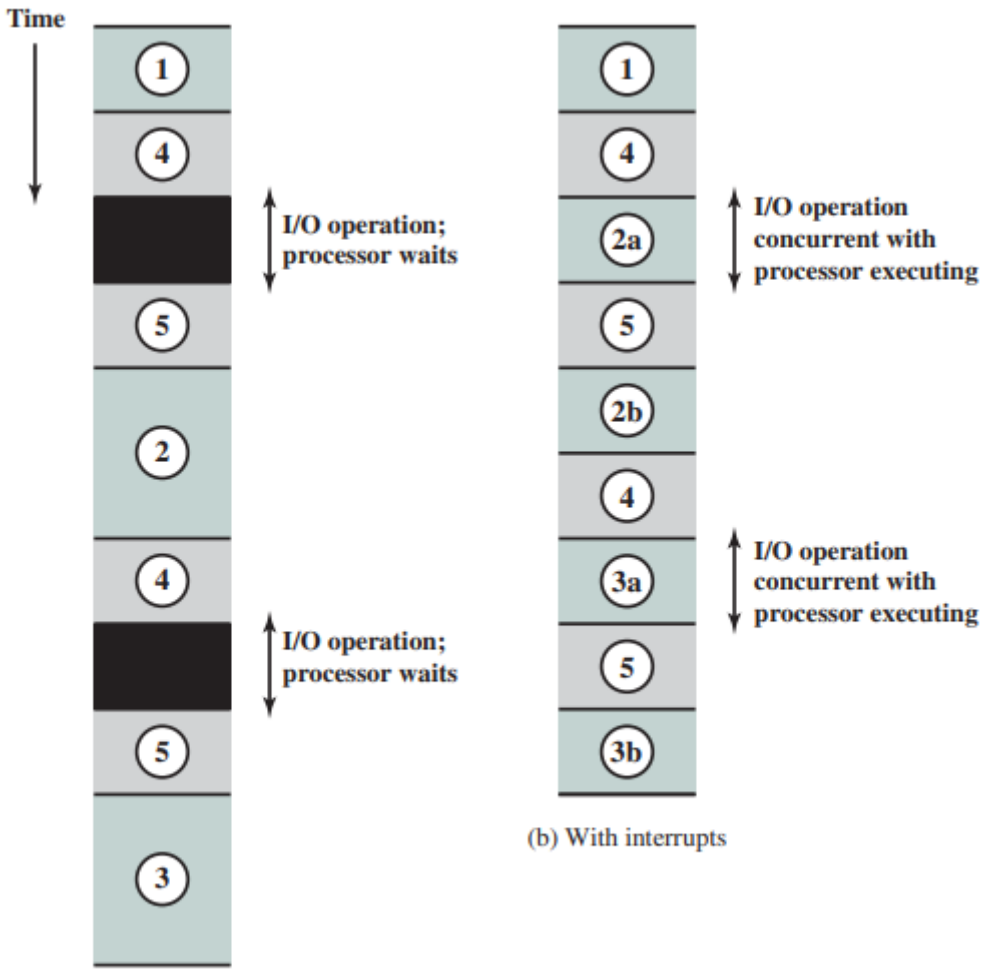
عادة ما يحدد هذا البرنامج طبيعة المقاطعة ويقوم بتنفيذها كافة الاجراءات المطلوبة.

عند اكتمال interrupt handler routine يمكن للمعالج استئناف تنفيذ الـ user program عند نقطة الانقطاع (point of interruption).

يجب تنفيذ تعليمات اضافية (Extra instructions) في الـ (interrupt handler) لتحديد طبيعة المقاطعة واتخاذ الاجراء المناسب.

ومع ذالك نظرا للكمية الكبيرة نسبيا من الوقت التي قد يتم اهدارها بمجرد انتظار الـ I/O يمكن استخدام المعالج بكفاءة اكبر باستخدام المقاطعات.

والصورة هاذي توضح.

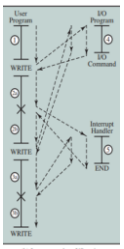


هذا مخطط توقيت يعتمد على تدفق التحكم (diagram based on the flow of control) .

كتل التعليمات البرمجية للبرنامج ملونه بالون الاخضر و I/O program code ملونه بالون الرمادي.

في صورة (a) يوضح هنا عدم وجود اي مقاطعات.

يجب ان ينتظر البرنامج تنفيذ عملية الـ I/O .



(b) Interrupts; short I/O wait

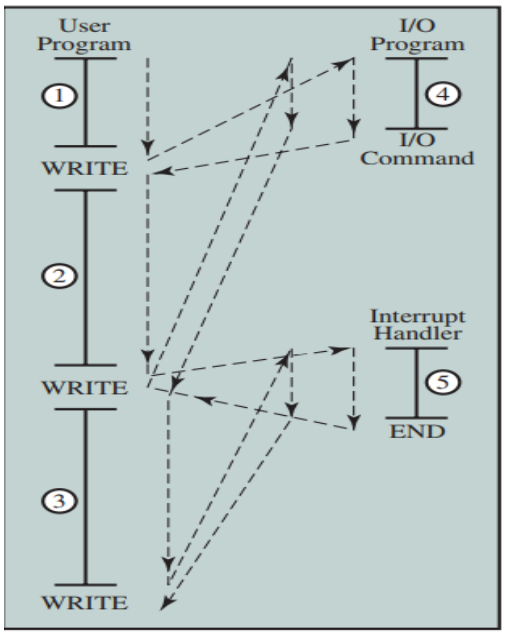
في (b) وهنا ماخوذة من الصورة التي قمنا بشرح عليها في الاعلى لـ write operation يفترضو بان الوقت المطلوب لعملية الـ I/O قصيره نسبيا بمعنى اقل من الوقت الازم لاستكمال تنفيذ التعليمات بين عمليات الكتابة في الـ User Program .

في هاذي الحالة تتم مقاطعة مقطع التعليمات البرمجية (segment of code) المسمى (code segment 2).

يتم تنفيذ جزء من الكود (2a) اثناء تنفيذ عملية الـ I/O ثم تحدث مقاطعة عند اكتمال عملية الـ I/O بعد المقاطعة يتم استئناف التنفيذ الجزء المتبقي من التعليمات (2b).

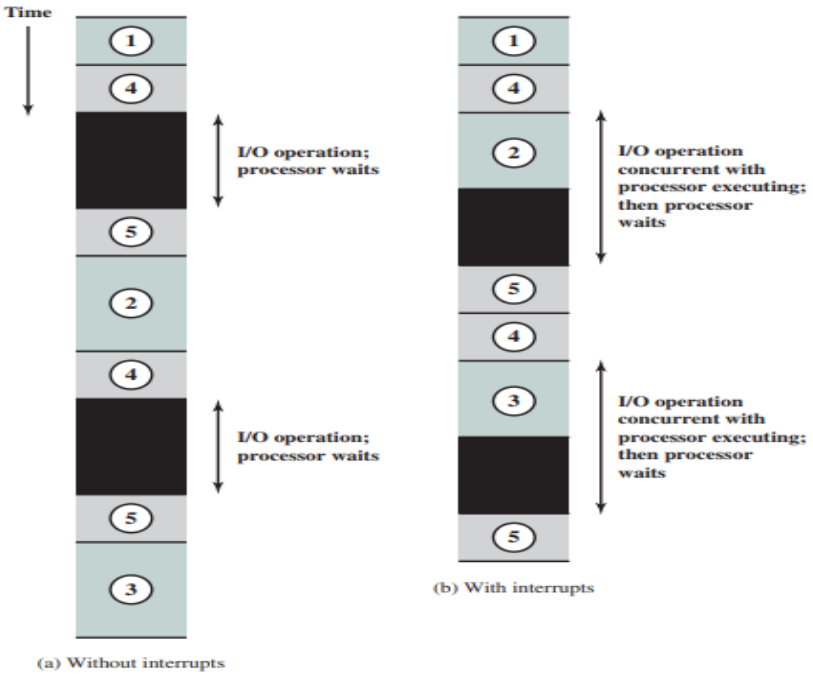
الحالة الاكثر شيوعيا خاصة بالنسبة للاجهزة البطيئة مثل الطابعات على سبيل المثال تنفيذ المقاطعة يعد اطول بكثير من تنفيذ سلسلة من التعليمات البرمجية للمستخدم.

مثال على ذالك هاذي الصورة :



(c) Interrupts; long I/O wait

وهاذي الصورة توضح بين البرنامج مع وبدون مقاطعات :



(a) Without interrupts

(b) With interrupts

يمكنك ان ترى يوجد مكاسب في الكفاءة لا الوقت الذي يجرى في عملية الـ i/o يتداخل تنفيذ التعليمات الـ user .

multiple interrupts

نحن ركزنا على عمل مقاطعة واحدة فقط.

لكن لنفترض ان من الممكن ان يحدث مقاطعات معددة. على سبيل المثال قد يتلقى البرنامج بيانات من خط الاتصال (communications line) ويطبع النتائج.

ستقوم الطابعة بانشاء مقاطعة في كل مرة تكمل فيها عملية طباعة. ستقوم وحدة التحكم في خط الاتصال (communication line controller) بانشاء مقاطعة في كل مره تصل فيها وحدة من البيانات.

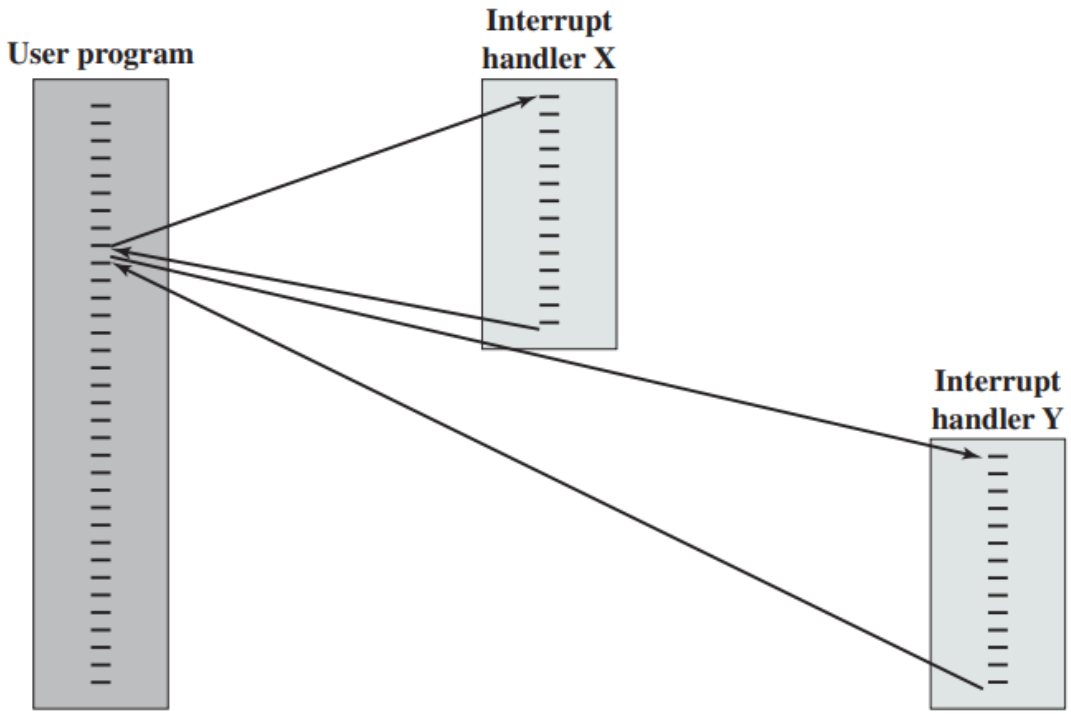
الوحدة (unit) ممكن ان تكون حرف او رقم او كتله هذا يعتمد على النظام الـ communications discipline .

يمكن اتباع طريقتين للتعامل مع المقاطعات المتعددة. الاول هو تعطيل المقاطعات (disable interrupts) اثناء معالجة المقاطعة . تعني المقاطعة المعطله (disable interrupts) ببساطة ان المعالج يمكنه تجاهل اشارة طلب المقاطعة هذه وسيقوم بذالك.

في حالة حدوث مقاطعة في هذا الوقت فانها تظل معلقة بشكل عام وسيتم فحصها بواسطة المعالج بعد ان يقوم المعالج بتشغيل المقاطعات (enabled interrupts).

وبتالي عند تنفيذ برنامج المستخدم (user program) وتحدث مقاطعة يتم تعطيل المقاطعات على الفور.

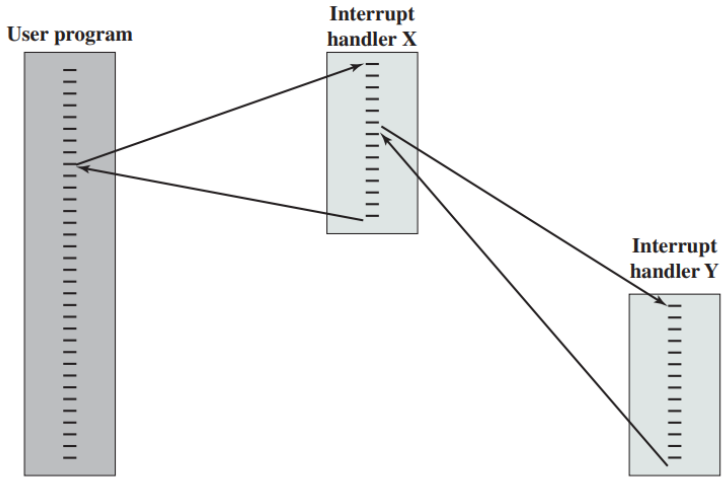
بعد اكتمال interrupt handler routine يتم تشغيل المقاطعات قبل استئناف برنامج المستخدم ويتحقق المعالج لمعرفة ما اذا كانت هناك مقاطعة اضافية قد حدثت.



(a) Sequential interrupt processing

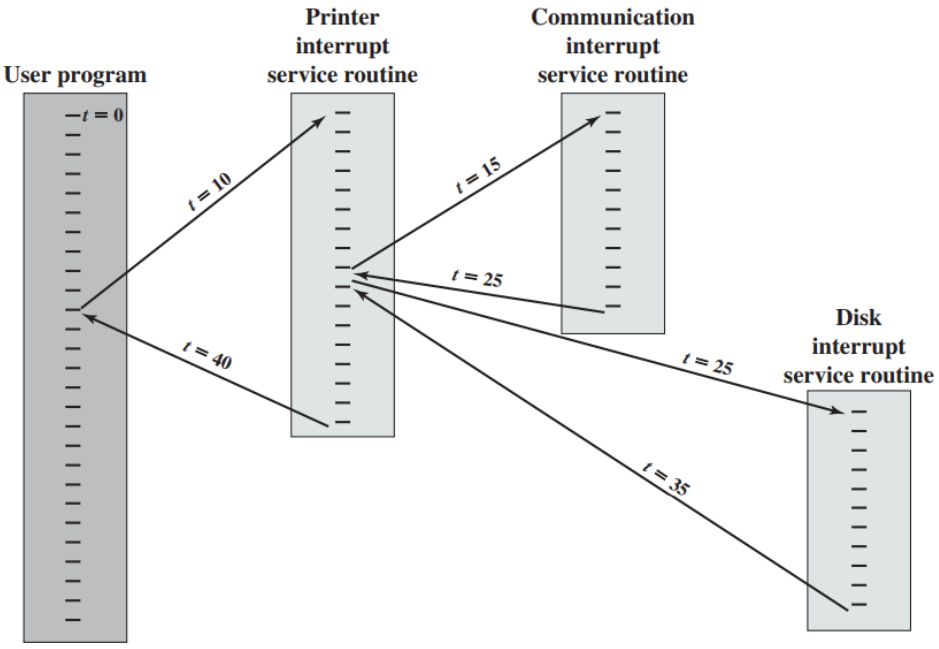
هاذي صورة توضح بشكل تسلسلي عمل المقاطعات.

الطريقة ثانية تحدد اولويات المقاطعة والسماح المقاطعة ذات الاولوية بالتسبب في مقاطعة الـ interrupt handler الاقل من ناحية الالولوية :



(b) Nested interrupt processing

وكمثال على هذا النهج الثاني فكر في نظام يحتوي على ثلاثة اجهزة i/o طابعة وقرص وخط اتصالات (printer, a disk, and a communications line) مع زيادة الاولويات بمقدار 2,4 & 5 على توالي.



هذا الشكل يوضح التسلسل.

عند t = 0 يبدأ الـ user program و عند t = 10 تحدث مقاطعة الطابعة. يتم وضع الـ user information على الـ system stack ويستمر في التنفيذ interrupt service routine (ISR) .

بينما لا يزال هذا الروتين في تنفيذ عند t = 15 تحدث مقاطعة (communications interrupt occurs).

نظرا لان خط الاتصالات (communications line) له اولوية اعلى من الطابعة يتم احترام المقاطعة.

الـ printer ISR صار لها مقاطعة ويتم دفع حالتها الى الـ stack ويستمر تنفيذ الـ communications ISR .

اثناء تنفيذ هذا الروتين تحدث مقاطعة عند القرص (disk interrupt occurs) (t = 20) .

هذه المقاطعة ذات أولوية أقل، يتم تعليقها ببساطة، ويتم تشغيل اتصالات ISR حتى الاكتمال.

عند اكتمال communications ISR (t = 25) تتم استعادة حالة المعالج السابقة وهي تنفيذ printer ISR .

ومع ذالك حتى قبل تنفيذ تعليمة واحدة في هذا الروتين يحترم المعالج المقاطعة القرص ذات الاولوية العليا ويتحكم في عمليات نقل الى disk ISR .

فقط عند اكتمال الـ (t = 35) يتم استئناف printer ISR .

وعند اكتمال t = 40 يعود التحكم اخيرا الى البرنامج والكل يعيش بسعاده المبرمج والمجتمع الكوكبي.

I/O Function

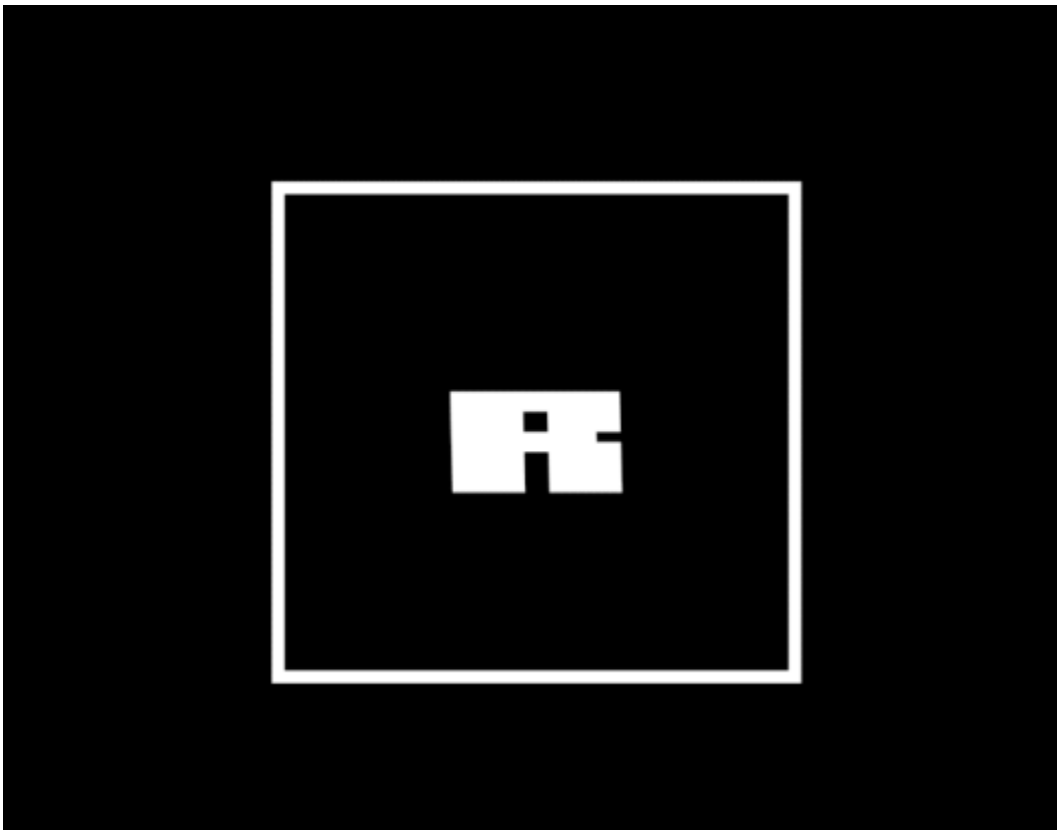
ناقشنا مواضيع شيقه منها المعالج و طريقة تفاعل بين الذاكرة والمعالج .

الـ I/O يحتاج الى فصل كامل لتقوم بفهمه فقط لكن ساقوم باعطائك بالمختصر المبسط ما هو و وظيفته.

الـ I/O module يمكن انها تتبادل معلومات بينها وبين المعالج مثال على ذالك الـ " disk controller" .

كمثل يمكن للمعالج يبدء القراءة والكتابة مع الذاكرة وتحديد عنوان موقع معين يمكن للمعالج ايضا قراءة بيانات من الـ I/O او الكتابة.

AhmadAlFareed



Twitter : https://twitter.com/dr_retkit

YouTube : https://www.youtube.com/@retkit1823