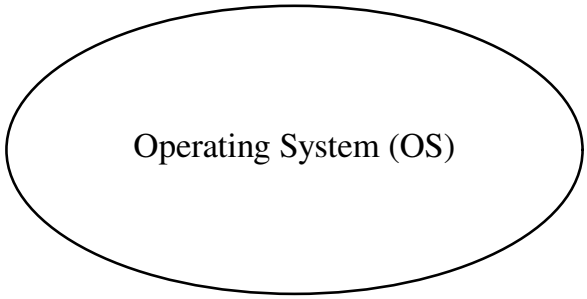
یادداشت‌های درس سیستم‌های عامل - بخش اول

اهداف این بخش از درس:

- چرا سیستم عامل؟
- وظایف اصلی سیستم عامل
- آشنایی فراخوانی‌های سیستمی
- آشنایی با سخت‌افزار

...


Operating System (OS)

...

...

...

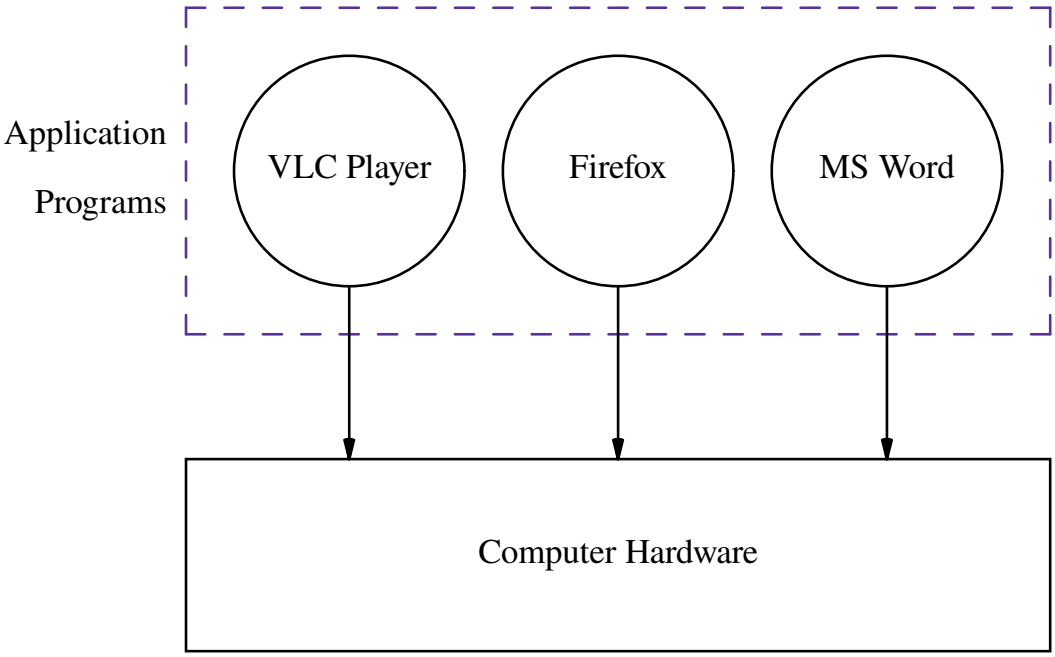

Computer Hardware

برنامه‌های کاربردی

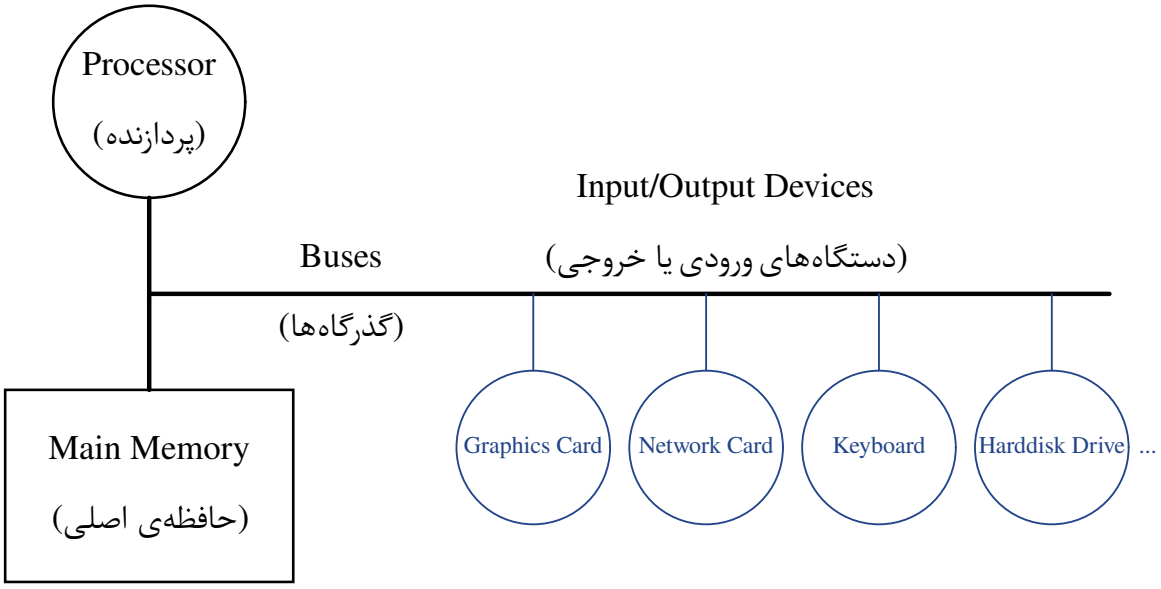
برنامه‌های کاربردی (Application Programs)

- برنامه‌های پردازشگر متن (Word Processor)
- برنامه‌های پخش موسیقی یا فیلم (Multimedia Player)
- بازی‌ها (Games)
- شبیه‌سازها (Simulators)
- مرورگر وب (Web Browsers)
- کامپایلرها (Compiler) و مفسرها (Interpreter)
- برنامه‌ی نمایش مستندهایی به فرمت‌های PDF و ...
- برنامه‌ی شرکت در کلاس‌های مجازی
- ماشین حساب و ...

برنامه‌های کاربردی + سخت‌افزار

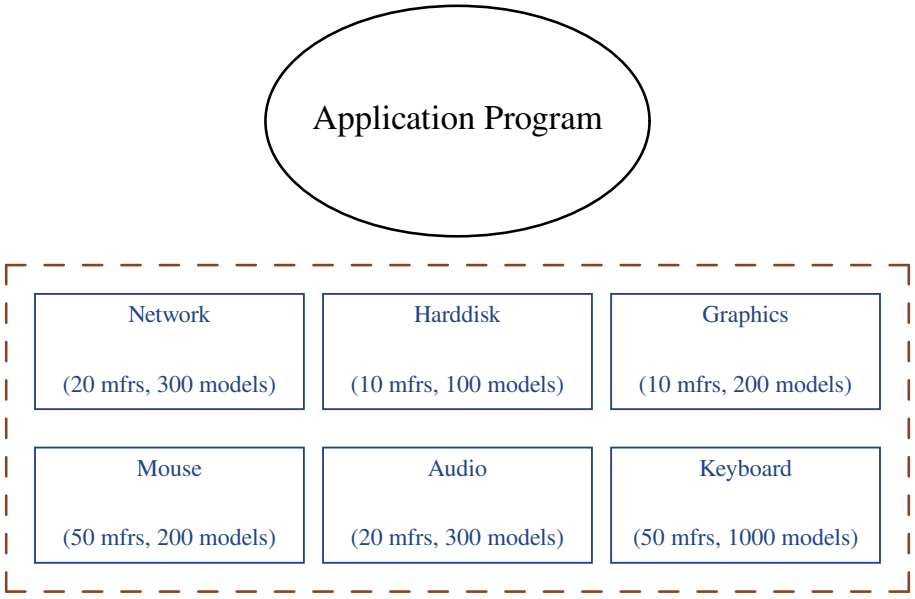


- برنامه‌های کاربردی مستقیما از سخت‌افزار استفاده کنند؟

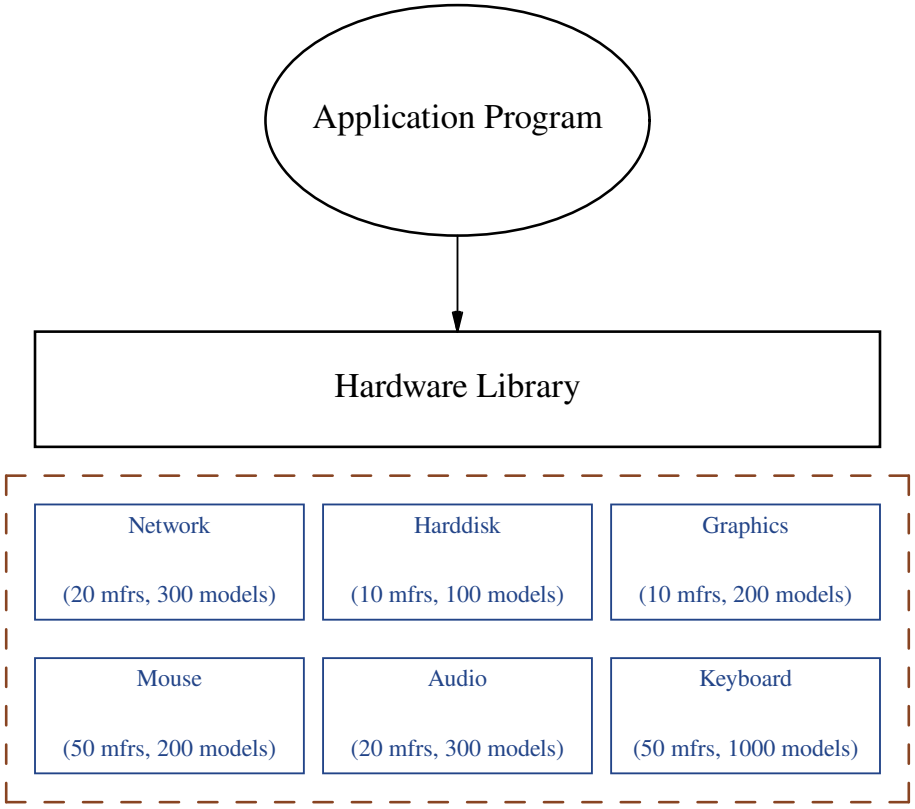

Processor
(پردازنده)
Input/Output Devices
(دستگاه‌های ورودی یا خروجی)
Buses
(گذرگاه‌ها)
Main Memory
(حافظه‌ی اصلی)
Graphics Card
Network Card
Keyboard
Harddisk Drive
...

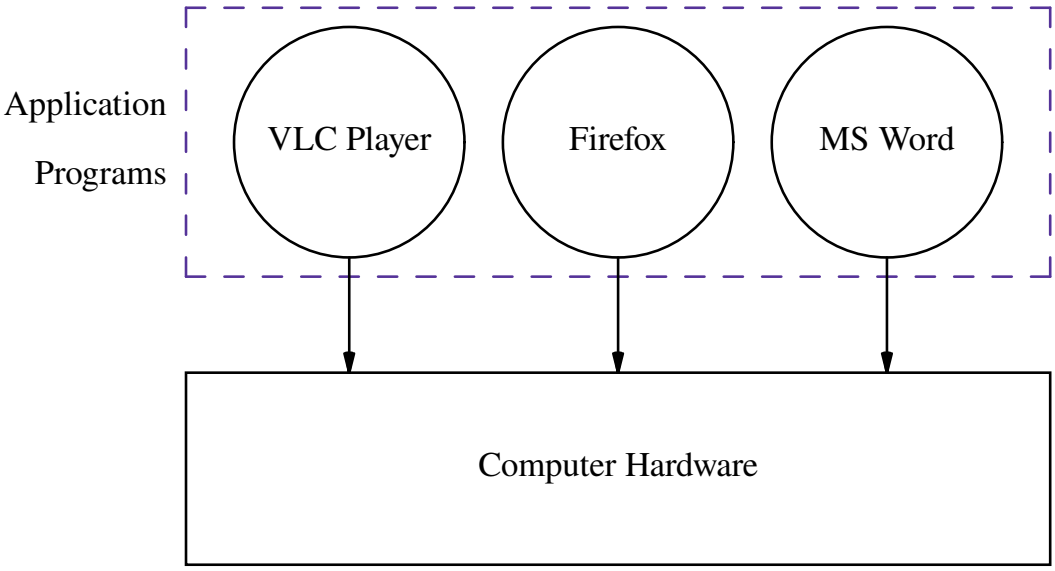
استفاده مستقیم برنامه‌ی کاربردی به سخت‌افزار

- تنوع قطعات سخت‌افزاری
- شرکت‌های سازنده + مدل‌های متنوع
- مستندهای طولانی برای هر قطعه



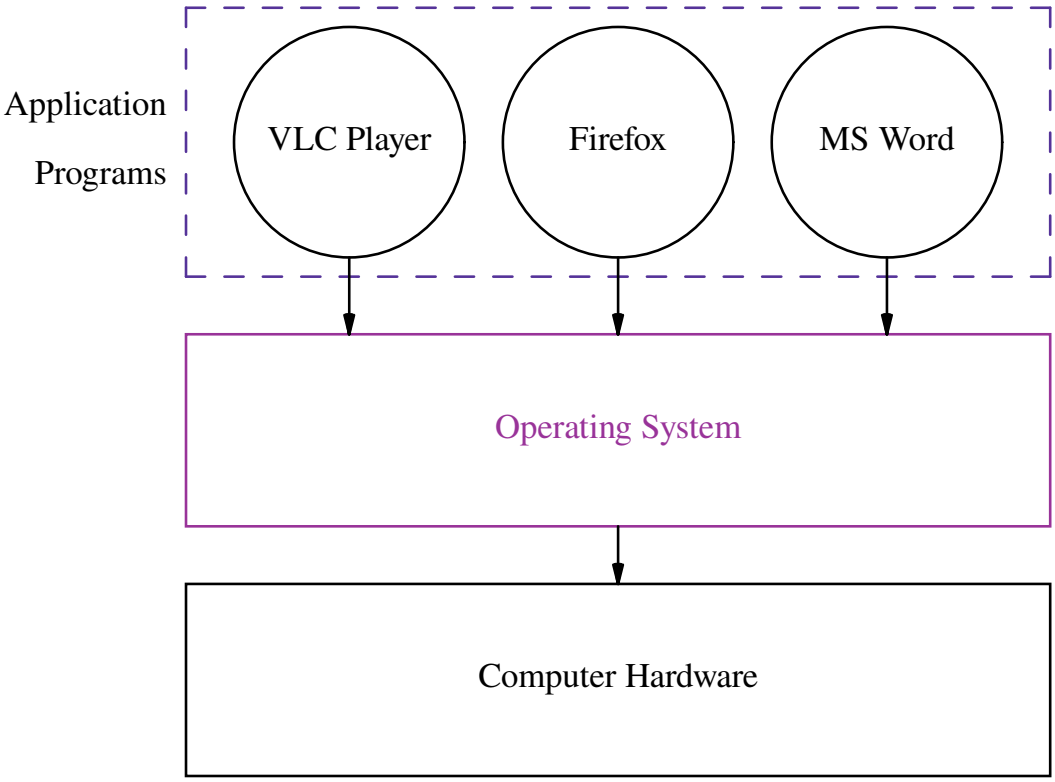
- قرار دادن ارتباط با سخت‌افزار در یک کتابخانه





نیازهای دیگر که کتابخانه برای آنها مناسب نیست:

- اجرای همزمان پردازه‌ها
- تقسیم منابع
- جلوگیری از تداخل برنامه‌ها
- تعامل بین پردازه‌ها


Application
Programs
VLC Player
Firefox
MS Word
Operating System
Computer Hardware

آیا همیشه سیستم عامل لازم هست؟

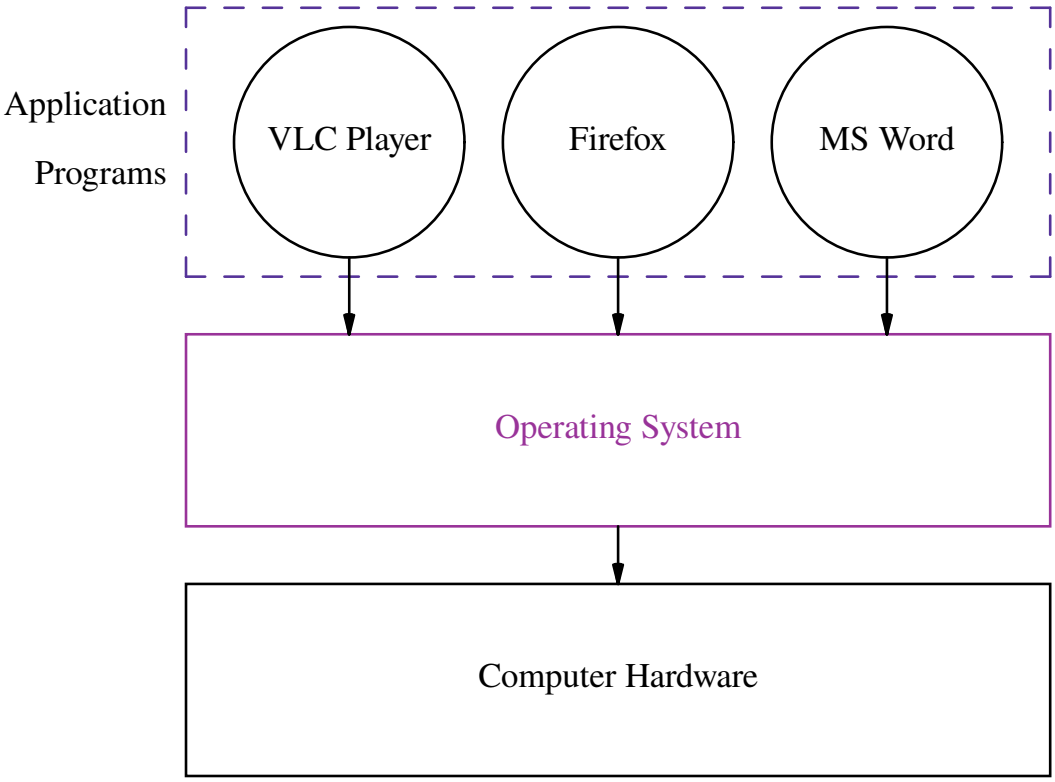
گاهی به یک سیستم عامل احتیاج نیست:

- فقط یک برنامه اجرا می‌شود
- سخت‌افزارها محدود هستند یا راه‌انداز آنها در یک کتابخانه موجود هست
- به خدمات دیگر سیستم عامل نیاز نیست

- Linux: هسته‌ی لینوکس (https://kernel.org/)

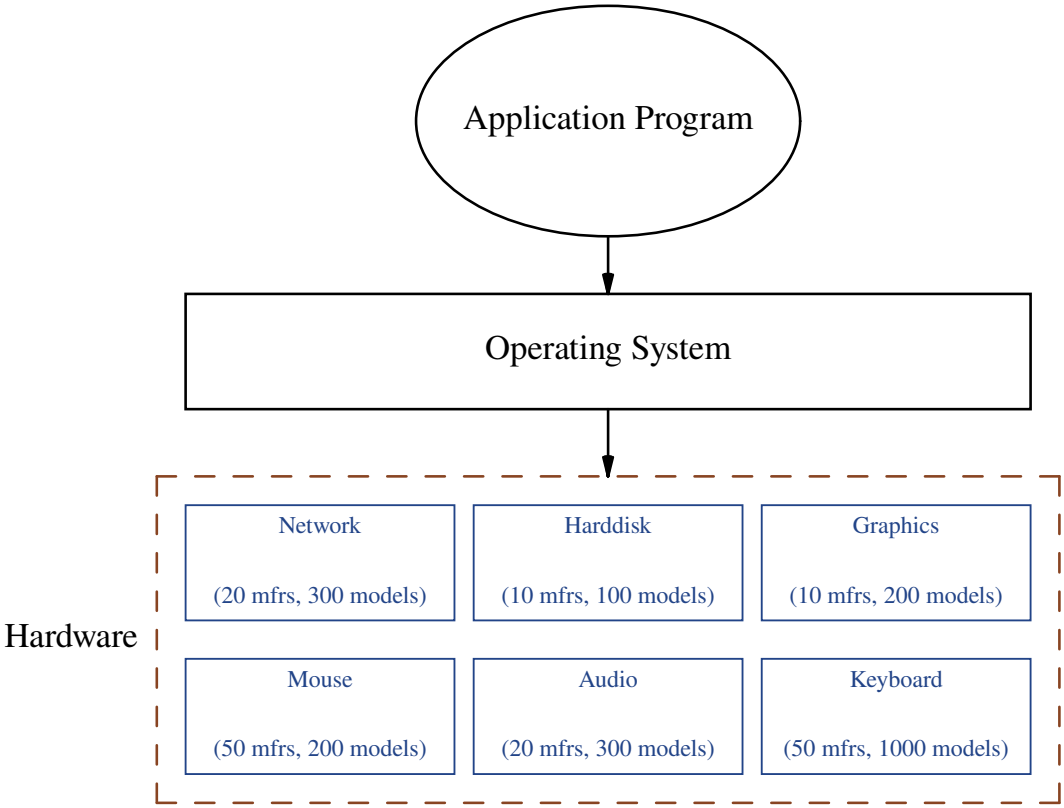
  شروع در ۱۹۹۱؛ بیشتر از ۳۳ میلیون خط کد
- FreeBSD: یکی از انواع BSD (Berkeley Software Distribution)؛

  برگرفته از Unix (https://www.freebsd.org/)
- Illumos: برگرفته از Solaris (https://www.illumos.org)



سیستم عامل مدیریت برنامه‌های در حال اجرا و سخت‌افزار را به عهده می‌گیرد.

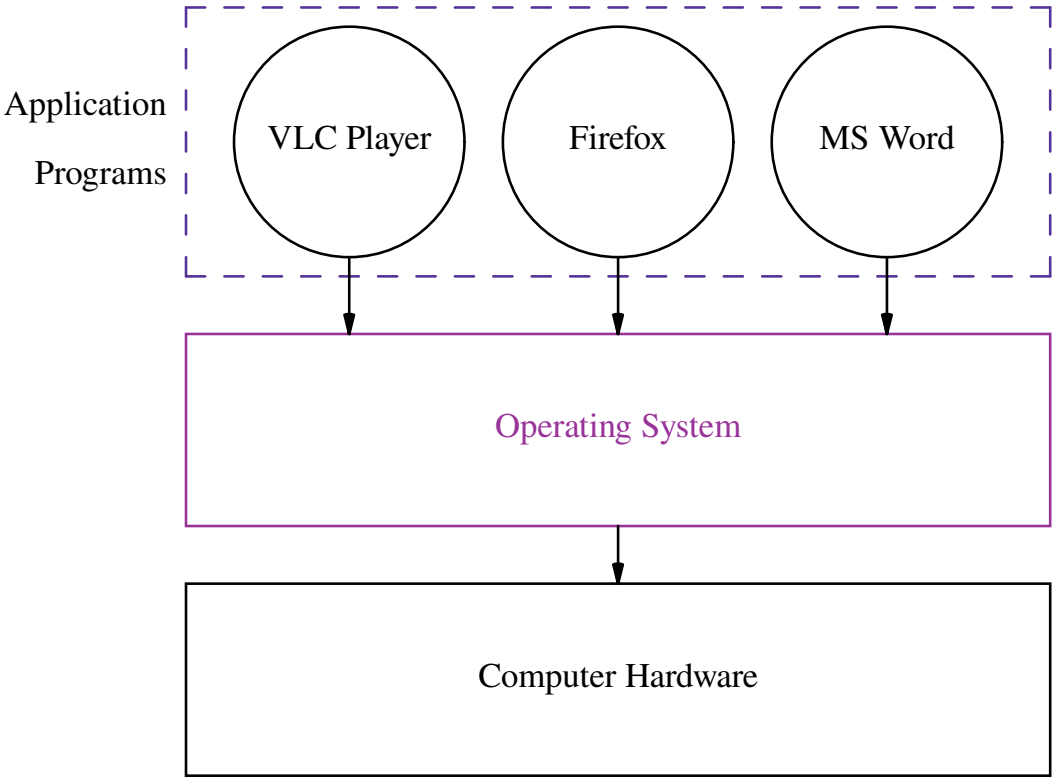
- سیستم عامل سخت‌افزار را مدیریت می‌کند.
- برنامه‌های در حال اجرا درخواست دسترسی به سخت‌افزار را به سیستم عامل می‌دهند.
- برای هر نوع قطعه‌ی سخت‌افزاری (مثل کارت گرافیک) یک رابط یکتا در اختیار پردازه‌ها قرار می‌دهد.



- برای برنامه‌ها نوع کارت صدا اهمیت چندانی ندارد، با کمک سیستم عامل از آن صدا پخش می‌شود.
- برای برنامه‌ها نوع دیسک اهمیت چندانی ندارد، با کمک سیستم عامل در آن اطلاعات ذخیره می‌شود و از آن بازیابی می‌شود.
- برای برنامه‌ها نوع کارت شبکه اهمیت چندانی ندارد، با کمک سیستم عامل در آن اطلاعات ذخیره می‌شود و از آن بازیابی می‌شود.

هر برنامه‌ی در حال اجرا از منابعی استفاده می‌کند.

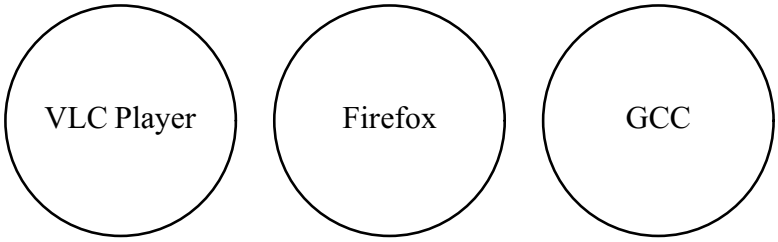
- منابعی مثل پردازنده، حافظه، شبکه، دیسک، چاپگر، گرافیک و ...



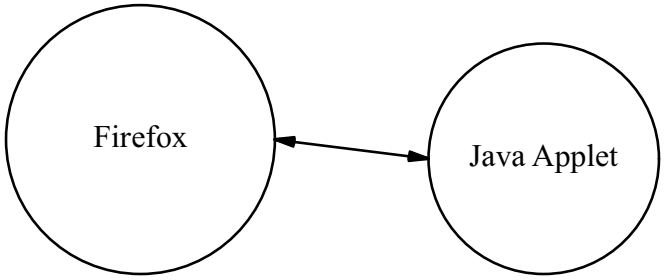
- سیستم عامل به شکلی این منابع را مدیریت می‌کند چند برنامه‌ی در حال اجرا به صورت اشتراکی بتوانند از آنها استفاده کنند.
- هر پردازه تصور می‌کند سخت‌افزار در اختیارش است.

مجزا نگه داشتن پردازه‌ها

- سیستم عامل هر پردازه را جدا نگاه می‌دارد.
- هر پردازه فقط به حافظه‌ای که به آن اختصاص داده شده دسترسی دارد.



- گاهی لازم است پردازه‌ها با هم تعامل کنند.



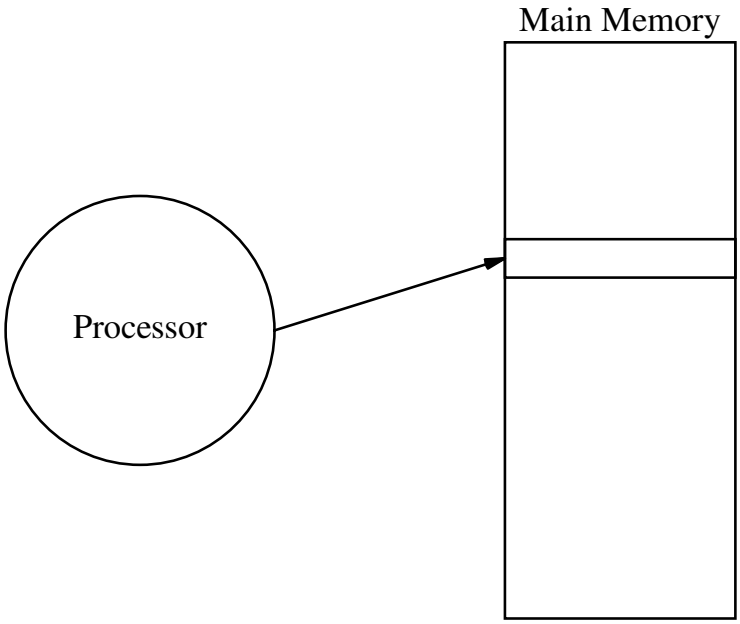
- سیستم عامل امکاناتی را برای ارتباط بین پردازه‌های مختلف فراهم می‌کند.

خلاصه

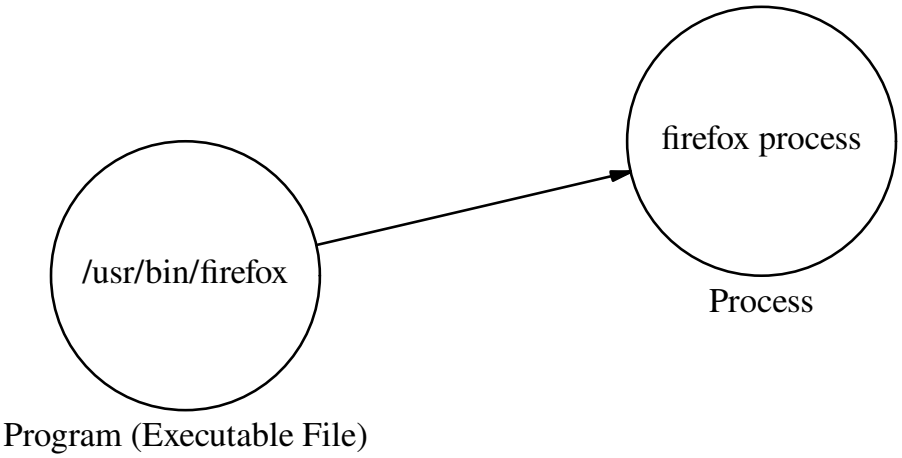
برخی از وظایف سیستم عامل

- انتزاع و مدیریت سخت‌افزار
- در اشتراک‌گذاری منابع بین پردازه‌ها
- جدا نگهداشتن پردازه‌ها
- تعامل پردازه‌ها

- در هر لحظه پردازنده دستوری از حافظه را اجرا می‌کند.

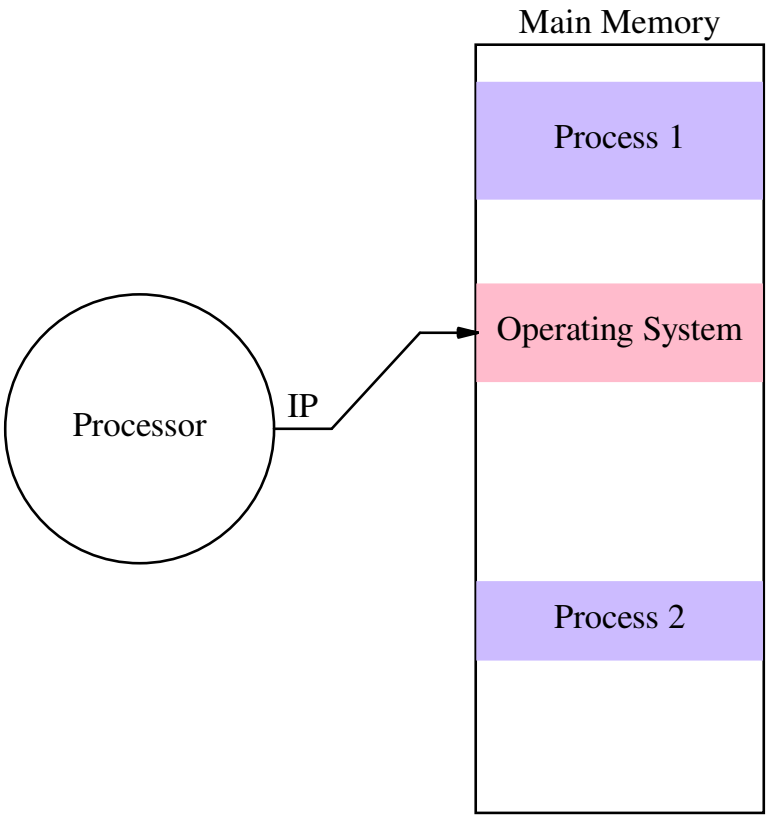


- یک برنامه (Program) یک فایل قابل اجرا است که در دیسک قرار دارد.
- برای نمونه فایل /usr/bin/firefox یک فایل اجرایی است.
- وقتی سیستم عامل شروع به اجرای یک برنامه می‌کند، قسمت‌هایی از آن را به حافظه‌ی اصلی انتقال می‌دهد.

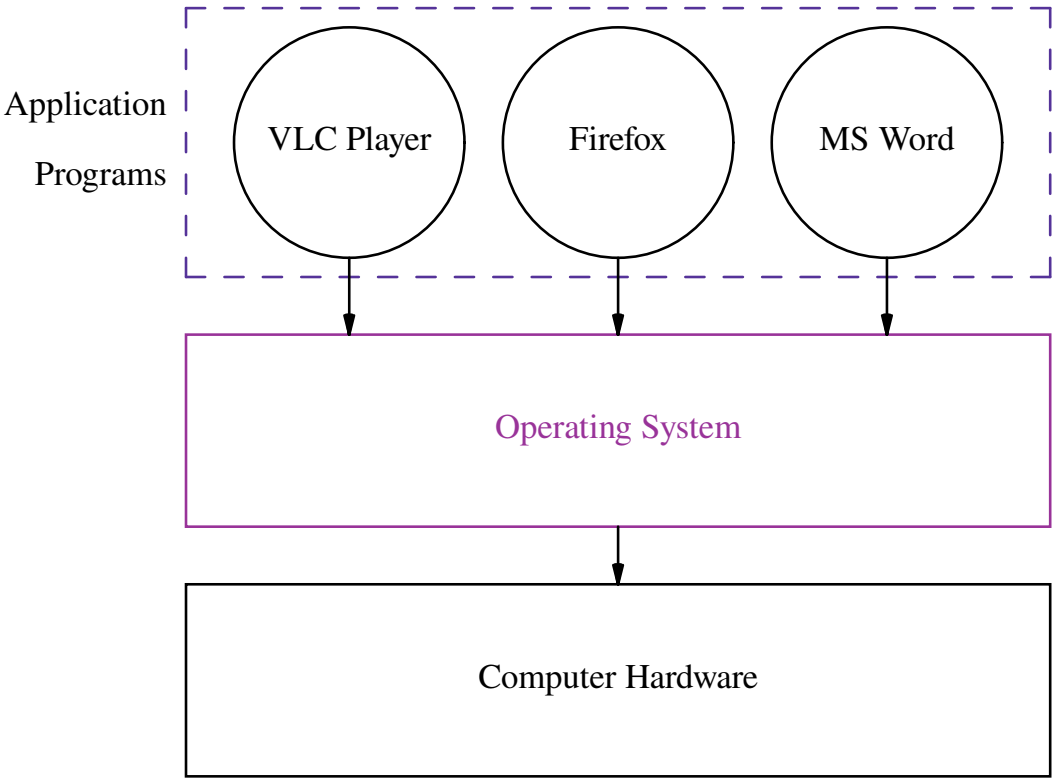


- به برنامه‌ی در حال اجرا Process، پردازه یا فرآیند گفته می‌شود.
- آیا ممکن است از یک برنامه چند پردازه موجود باشد؟

- در هر لحظه هر پردازنده دستوری از قسمتی از حافظه را اجرا می‌کند.
- این قسمت گاهی مربوط به برنامه‌های کاربردی و گاهی مربوط به سیستم عامل هست.

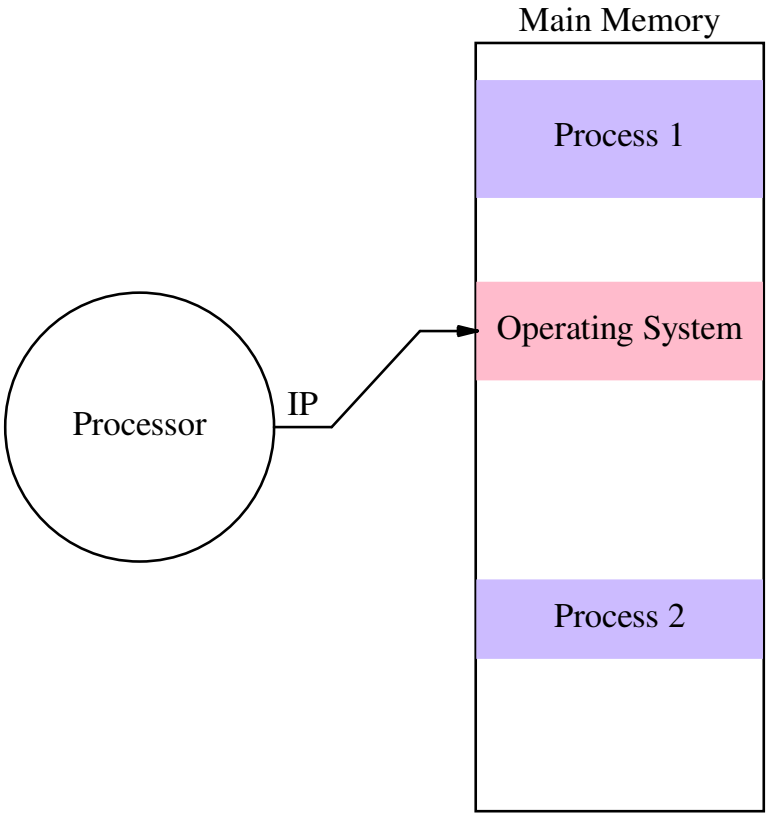


- اگر پردازه‌ها بتوانند مستقیما به سخت‌افزار دسترسی داشته باشند چه می‌شود؟

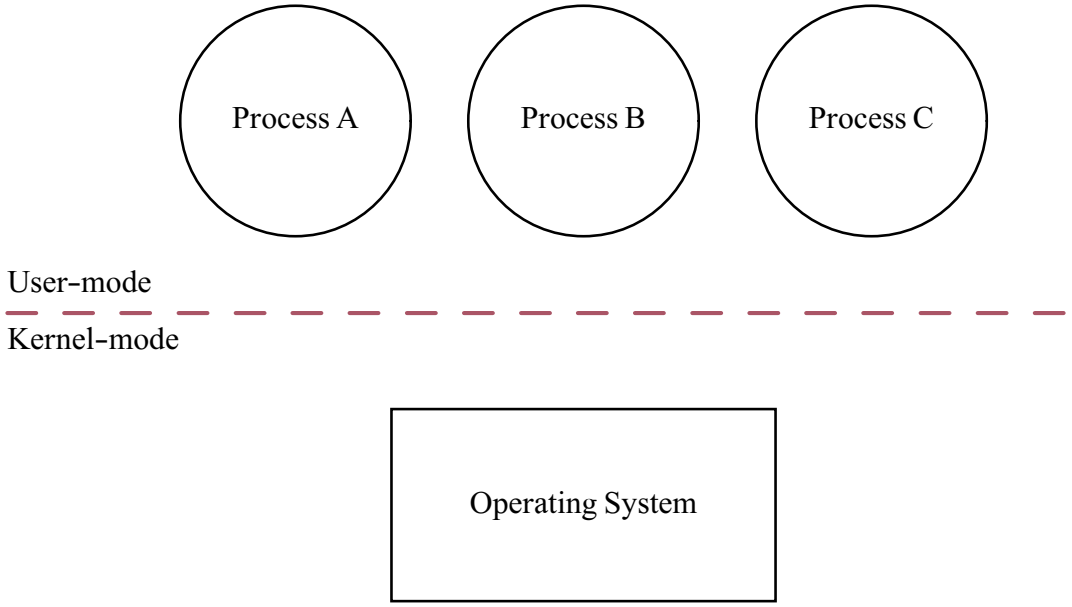


- چگونه می‌توان دسترسی پردازه‌ها را محدود کرد؟

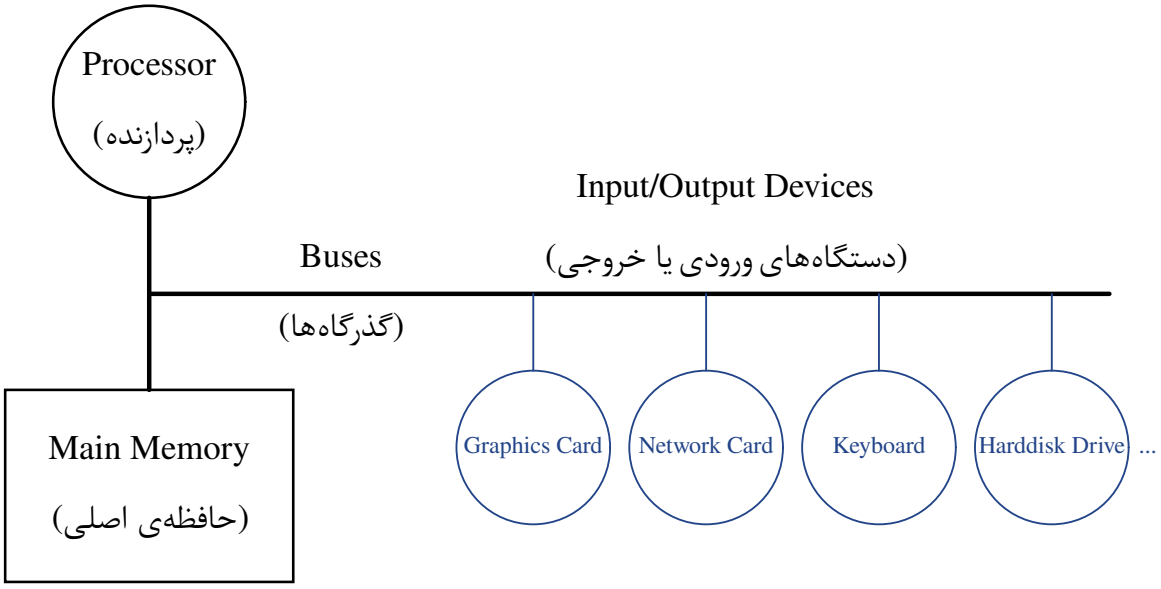
- پردازنده‌های امروزی در چند حالت اجرا می‌شوند.
- حالت کاربر (حالت محدود، User-mode): اگر پردازنده در این حالت باشد دستوراتی که به پردازنده دسترسی دارند را نمی‌تواند اجرا کند.
- حالت هسته (Kernel-mode، Supervisor-mode، System-mode، Priviledged-mode): اگر پردازنده در این حالت باشد می‌تواند بدون محدودیت به سخت‌افزار دسترسی داشته باشد؛ همه‌ی دستورات را می‌تواند اجرا کند.
- معمولا پردازنده‌ها حالت‌های بیشتری را پشتیبانی می‌کنند...



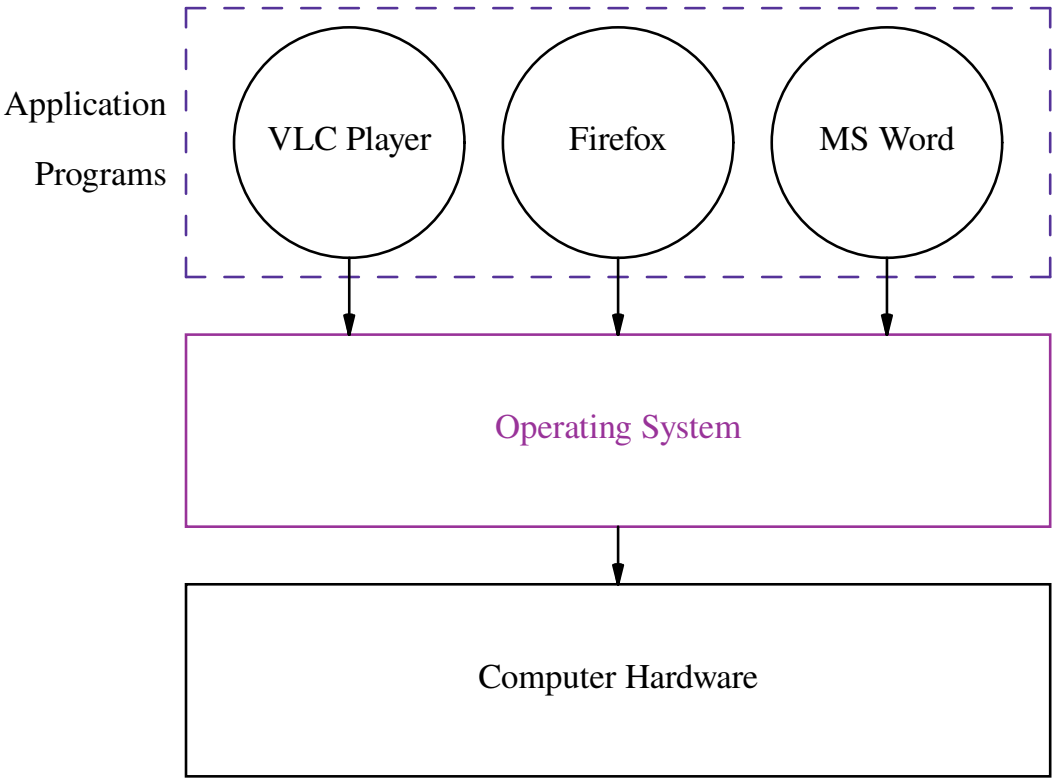
- پردازه‌ها در حالت کاربر اجرا می‌شوند.
- سیستم عامل در حالت هسته اجرا می‌شود.



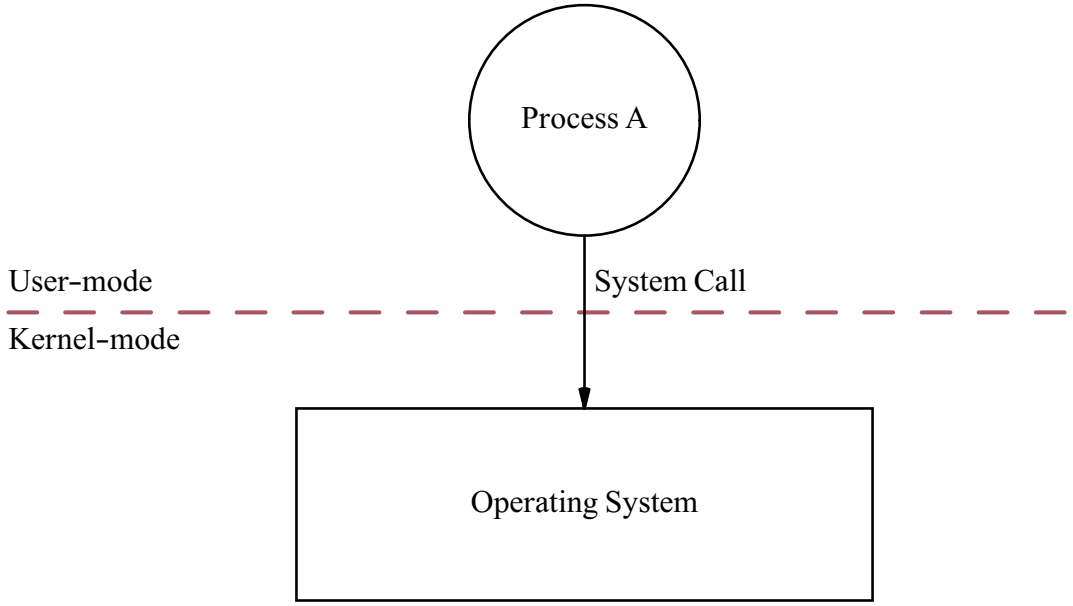
- به قسمتی که در حالت هسته اجرا می‌شود، هسته یا Kernel هم گفته می‌شود.


Processor
(پردازنده)
Input/Output Devices
(دستگاه‌های ورودی یا خروجی)
Buses
(گذرگاه‌ها)
Main Memory
(حافظه‌ی اصلی)
Graphics Card
Network Card
Keyboard
Harddisk Drive
...

- پردازه‌ها چگونه به سخت‌افزار دسترسی داشته باشند؟



- فراخوانی سیستمی (System Call): درخواست‌های پردازه از سیستم عامل

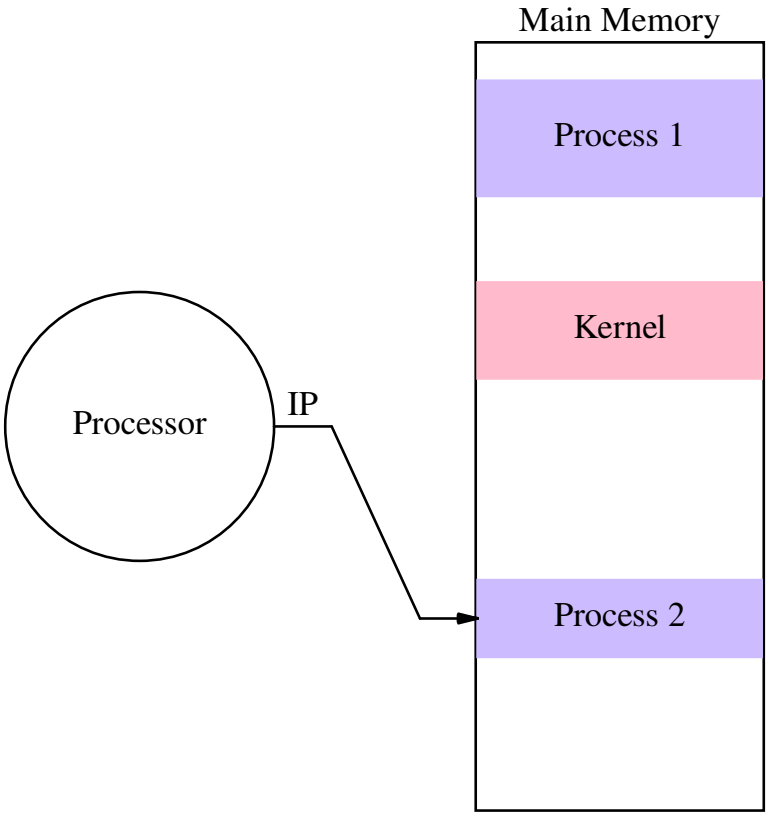


- برای اجرای فراخوانی سیستمی: وضعیت اجرای پردازنده از حالت کاربری به حالت هسته تغییر می‌کند و هسته اجرا می‌شود.

برای بسیاری از خدمات سیستم عامل، فراخوانی سیستمی وجود دارد.

- برای دسترسی به فایل‌ها: open()، read()، write()، lseek() و ... .
- برای مدیریت پردازه‌ها: fork()، exec()، wait() ...
- سیگنال‌ها: signal()، kill() و ...
- مدیریت فایل‌سیستم: chown، chmod، unlink() و ...
- ارتباط بین پردازه‌ای
- ...

- پردازنده در هر لحظه در حال اجرای قسمتی از حافظه است.

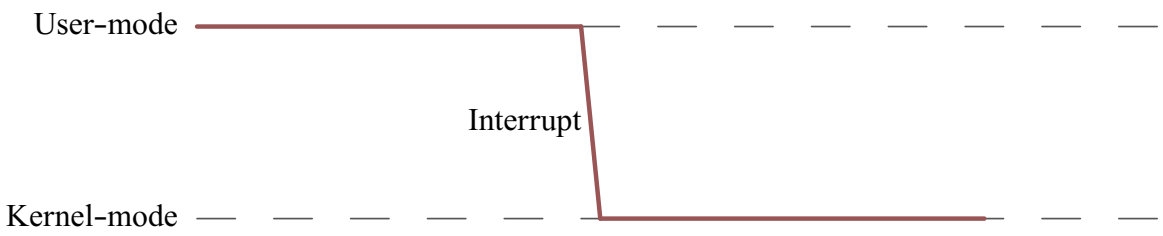


- فرض کنید پردازنده در حالت کاربر و در حال اجرای پردازه‌ای باشد.
- وقفه (Interrupt): توقف اجرای پردازنده و انتقال به حالت هسته و اجرای قسمت مشخصی از هسته.

چرا وقفه

در چه صورتی وقفه لازم است؟

- برای درخواست از سیستم عامل؟
- برای سخت‌افزار؟
- برای وضعیت خطا؟



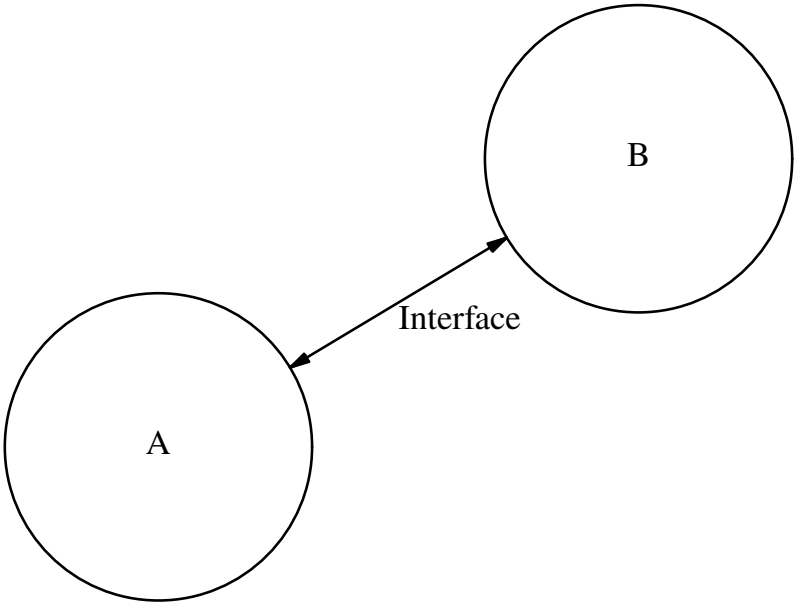
انواع وقفه

- وقفه‌های سخت‌افزاری (Hardware Interrupts): رخدادهای سخت‌افزاری.
- فراخوانی‌های سیستمی (System Calls): درخواست‌های پردازه‌ها از سیستم عامل (قبلا آشنا شدیم).
- خطاهای نرم‌افزاری (Software Traps): در مورد حافظه یا عملیات حسابی.

- چرا سیستم عامل؟
- وظایف سیستم عامل
- چرا چند حالت در پردازنده‌ها؟
- فراخوانی سیستمی
- وقفه‌ها

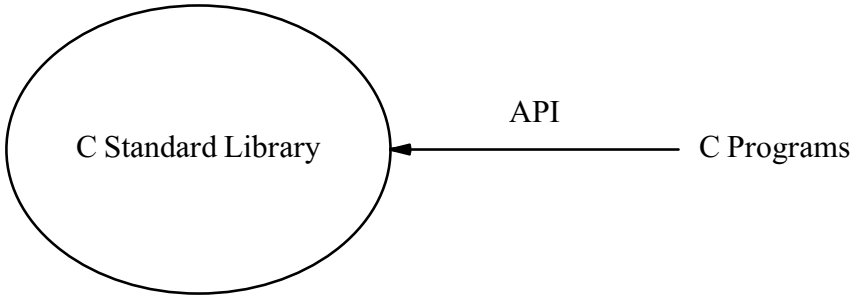
- برنامه‌های زیادی نوشته‌اید.
- آیا فراخوانی سیستمی انجام داده‌اید؟

- رابط (Interface): روش انتقال اطلاعات بین دو قسمت از یک سیستم



- قسمت‌ها می‌توانند نرم‌افزاری، سخت‌افزاری، دستگاه‌هایی که به کامپیوتر وصل می‌شوند و حتی انسان باشند.
- مثال: Touchscreen، موشواره، صفحه‌کلید، کابل USB و GUI (Graphical User Interface).

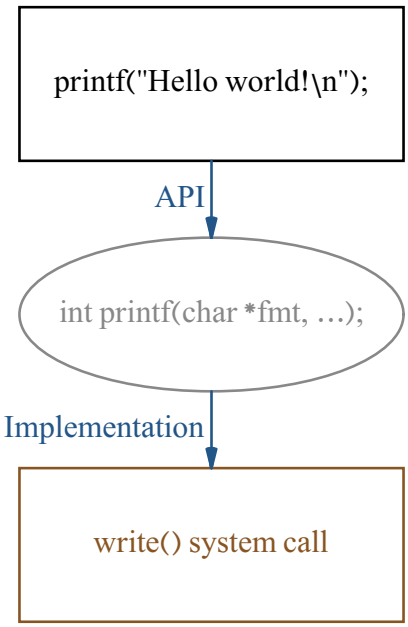
- برای استفاده از یک کتابخانه، از توابع آن استفاده می‌کنیم.
- این توابع جزء API (Application Programming Interface) کتابخانه هستند.



- مثال: API کتابخانه‌ی استاندارد زبان‌های مختلف
- مثال: API کتابخانه‌ی GTK یا ...

استفاده از API به جای فراخوانی سیستمی

- معمولا شما از API کتابخانه‌ها به جای فراخوانی مستقیم فراخوانی‌های سیستمی استفاده می‌کنید.
- استفاده از API قابلیت انتقال (Portability) بیشتری دارد. چرا؟
- استفاده از API معمولا ساده‌تر است. چرا؟
- در پیاده‌سازی توابع API ممکن است از فراخوانی سیستمی استفاده شود.

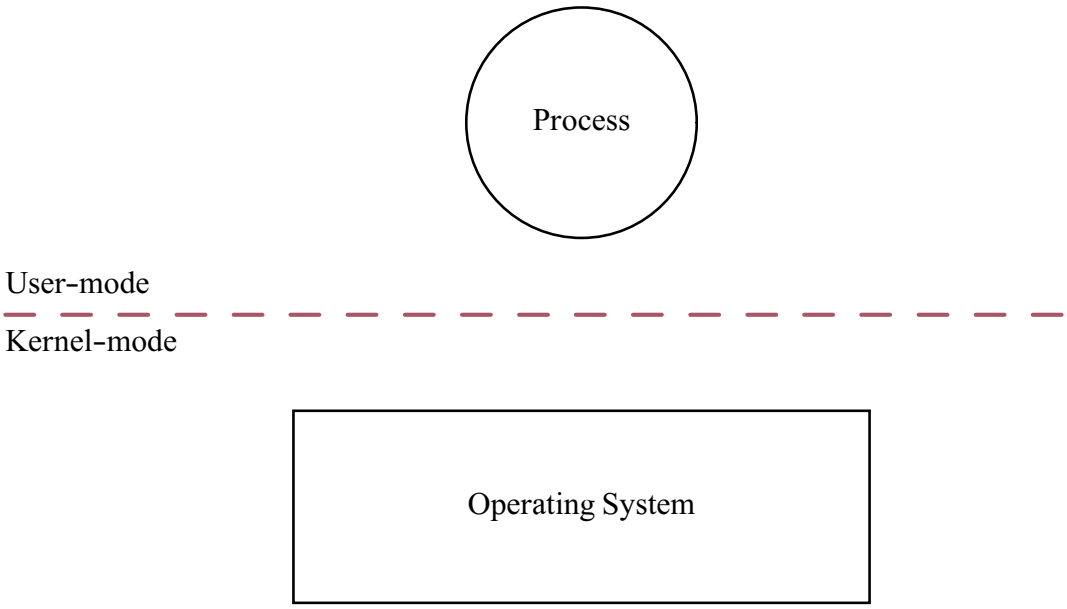


- تعداد فراخوانی‌های سیستمی که یک پردازه معمولا انجام می‌دهد بسیار زیاد است.

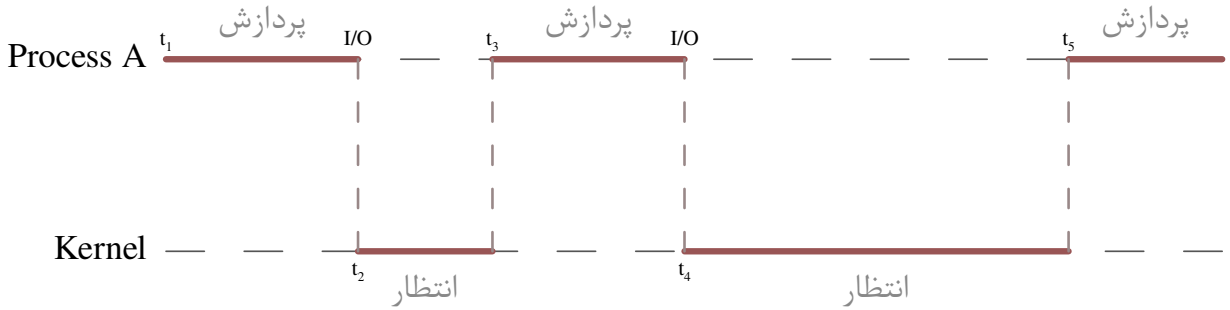
```
$ strace ls
execve("/root/b/ls", ["ls"], [/* 24 vars */]) = 0
arch_prctl(ARCH_SET_FS, 0x7ffdf8765030) = 0
geteuid()                               = 0
getegid()                               = 0
ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
ioctl(1, TIOCGWINSZ, {ws_row=28, ws_col=97, ws_xpixel=0, ws_ypixel=0}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f...
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f...
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f...
stat(".", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=80, ...}) = 0
access(".", F_OK)                       = 0
open(".", O_RDONLY|O_DIRECTORY)         = 3
fcntl(3, F_SETFD, FD_CLOEXEC)           = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f...
getdents(3, /* 4 entries */, 4083)      = 104
getdents(3, /* 0 entries */, 4083)      = 0
close(3)                                = 0
munmap(0x7f14c3452000, 4096)            = 0
lstat("log", {st_mode=S_IFREG|0644, st_size=1061, ...}) = 0
write(1, "log0, 4)                    = 4
exit(0)                                 = ?
+++ exited with 0 +++
```

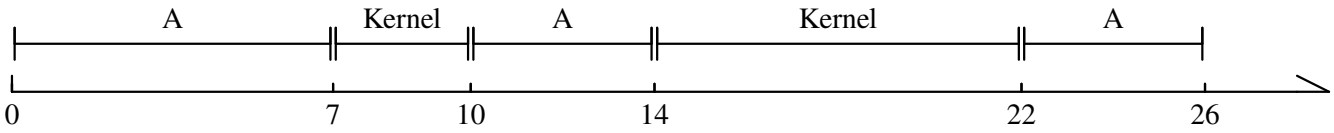
- سیستم عامل: پردازنده به کدام پردازه داده شود؟

- فقط یک پردازه در هر لحظه سیستم عامل در حال اجرا است.
- مثال: Microsoft DOS

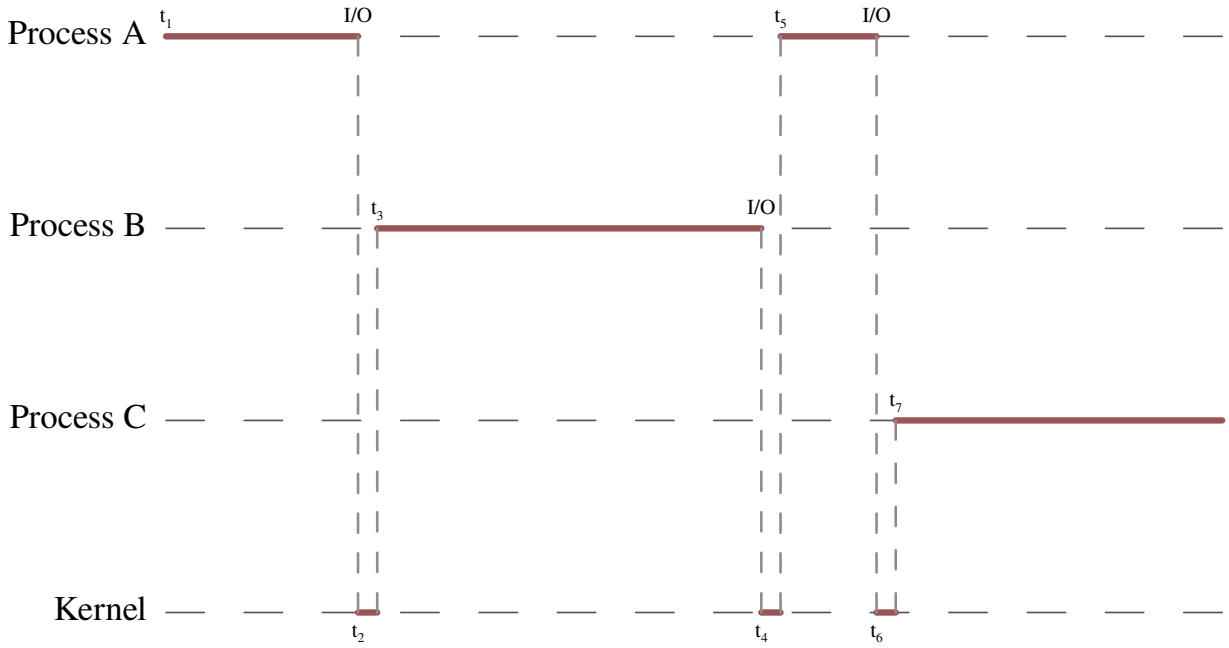


- اگر پردازه عمل ورودی یا خروجی انجام دهید، پردازنده بیکار می‌ماند.
- مثل: خواندن از صفحه‌کلید.
- پردازنده بیکار می‌ماند.





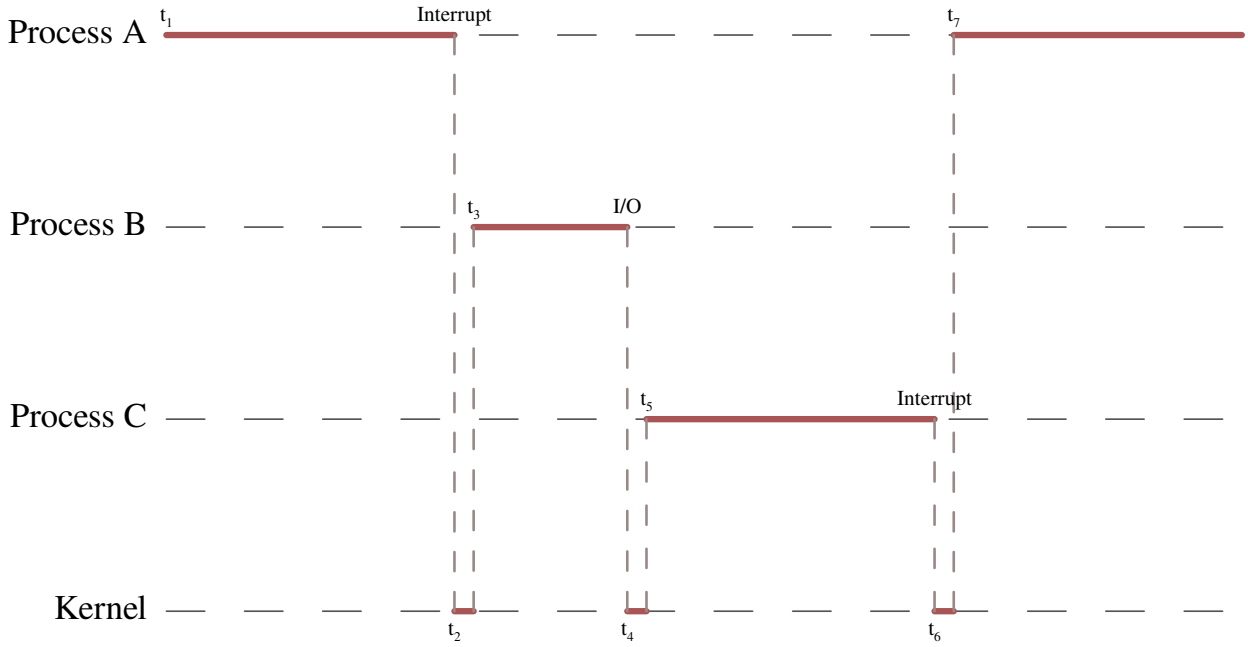
- چند برنامگی (Multiprogramming): چند برنامه اجرا می‌شوند.



- اگر پردازه‌ای منتظر شود، سیستم عامل یک پردازه‌ی آماده‌ی اجرا را اجرا می‌کند.
- درجه‌ی چند برنامگی (Degree of Multiprogramming): تعداد پردازه‌های در حال اجرا.

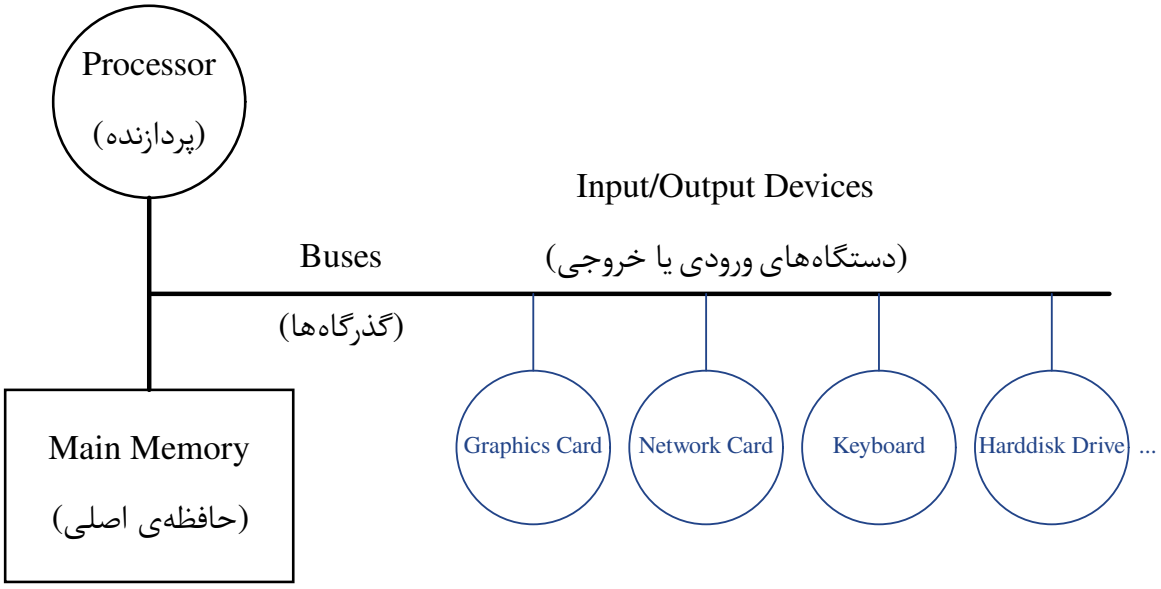
- مناسب برای وظیفه‌های دسته‌ای و غیر محاوره‌ای (Non-interactive)
- در برنامه‌های محاوره‌ای چه مشکلی رخ می‌دهد؟
- زمانبندی مبتنی بر همکاری (Cooperative Scheduling)

- سیستم‌های عامل اشتراک زمانی (Time-sharing)
- برای هر پردازه حداکثر زمان پردازش تعیین می‌شود.
- اگر پردازه پس از مقدار تعیین شده عملیات ورودی و خروجی انجام نداد، پردازنده به زور گرفته می‌شود.

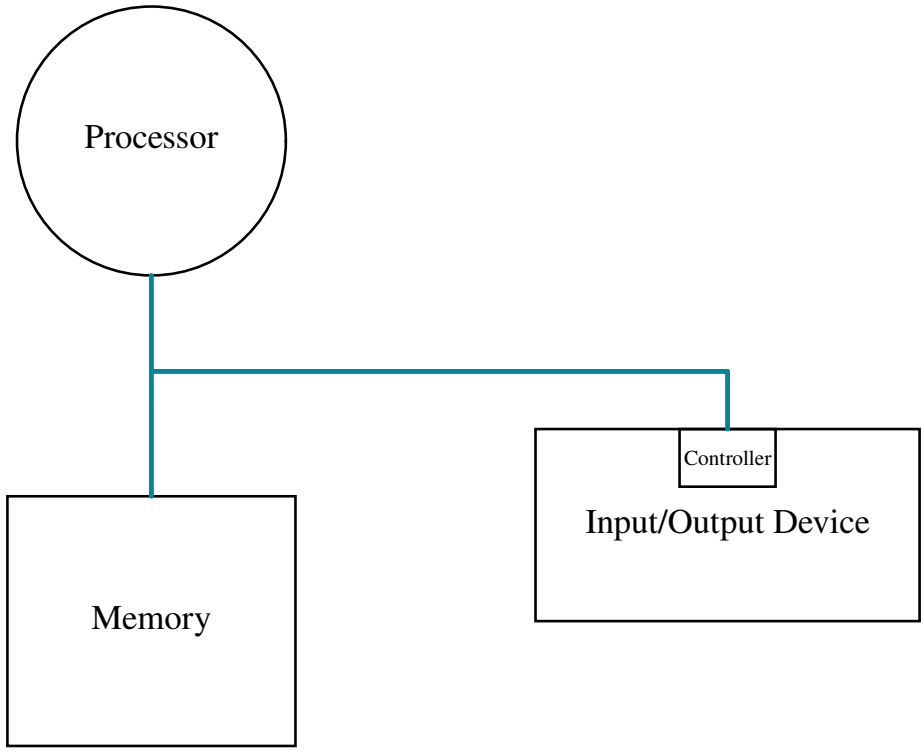


- چگونه پرداز پردازنده قطع شود؟

- سخت‌افزاری به نام Timer وقفه‌ی سخت‌افزاری ایجاد می‌کند.
- زمان وقفه‌ها قابل تنظیم است.

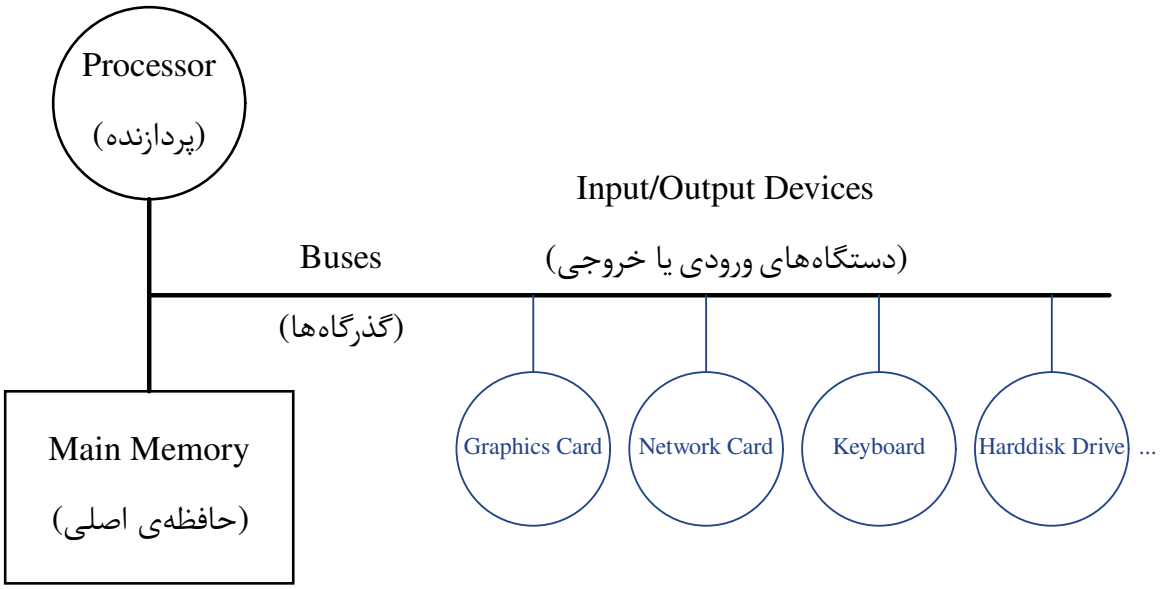

Processor
(پردازنده)
Input/Output Devices
(دستگاه‌های ورودی یا خروجی)
Buses
(گذرگاه‌ها)
Main Memory
(حافظه‌ی اصلی)
Graphics Card
Network Card
Keyboard
Harddisk Drive
...

- راه‌انداز (Driver؛ به صورت کامل Device Driver): قسمتی از سیستم عامل که با سخت‌افزار (دستگاه ورودی یا خروجی) در ارتباط است.
- راه‌انداز دستگاه را مقدار دهی می‌کند، به آن درخواست می‌فرستد، وضعیت آن را کنترل می‌کند و پاسخ‌های آن را دریافت می‌کند.
- در هر دستگاه ورودی یا خروجی قسمتی به نام Controller وجود دارد که با راه‌انداز در ارتباط است.



- رخدادهایی در دستگاه ورودی یا خروجی رخ می‌دهند که سیستم عامل باید از آنها آگاه شود.
- به طور کلی دو روش برای این کار وجود دارد.
- با یکی از آنها قبلا آشنا شده‌ایم.

- الف) وقفه‌های سخت‌افزاری (Interrupt-Driven I/O): در صورت بروز رخداد، با وقفه‌ی سخت‌افزاری دستگاه سیستم عامل را آگاه می‌سازد.
- ب) سرکشی (Polling-Based I/O): سیستم عامل به صورت متناوب وضعیت دستگاه‌های ورودی یا خروجی را بررسی می‌کند.



- بیشتر سیستم‌های عامل امروزی از روش مبتنی بر وقفه استفاده می‌کنند. چرا؟
- اما در شرایطی ممکن است برای برخی از دستگاه‌های ورودی یا خروجی به صورت موقتی به سرکشی روی بیاورند.
- در چه شرایطی سرکشی عملکرد بهتری دارد؟

- دو روش کلی برای انتقال اطلاعات بین دستگاه ورودی یا خروجی و حافظه‌ی اصلی وجود دارد.
- الف) مبتنی بر درگاه (Port-Based I/O):

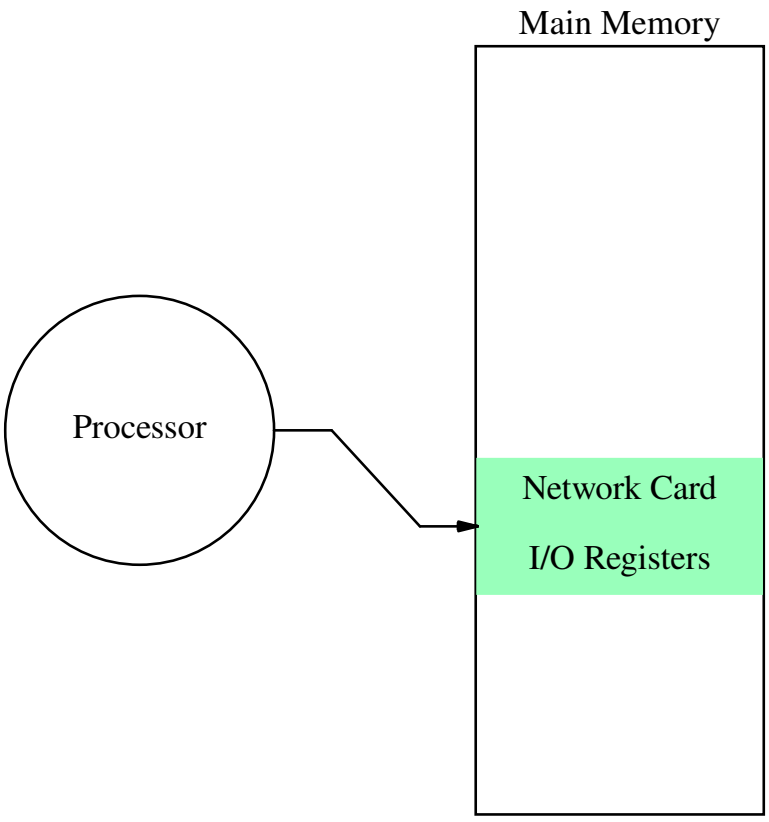
به دستگاه‌ها یک درگاه از پردازنده اختصاص می‌یابد.

با دستورات خاصی می‌توان از درگاه‌ها خواند یا نوشت.

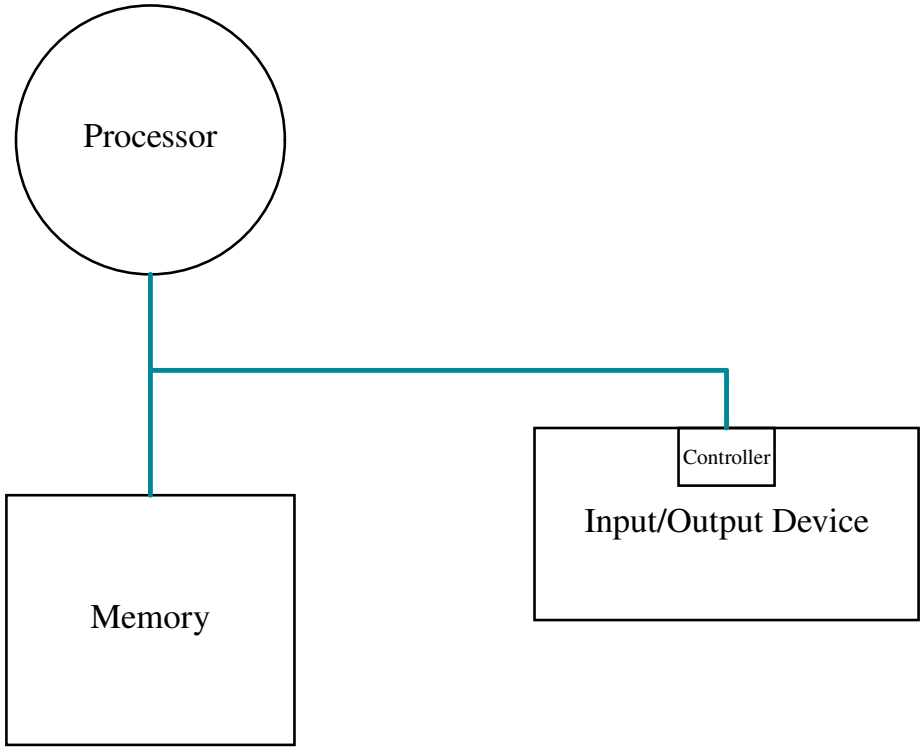
مثال: در PC درگاه صفحه‌کلید 0x60 هست:

```
# Read from PS/2 keyboard
in 0x60, al
# Write to timer register
out al, 0x43
```

- ب) نگاشت شده به حافظه (Memory-Mapped I/O):
- رجیسترهای دستگاه‌های ورودی یا خروجی به آدرس‌های حافظه نگاشت می‌شوند.



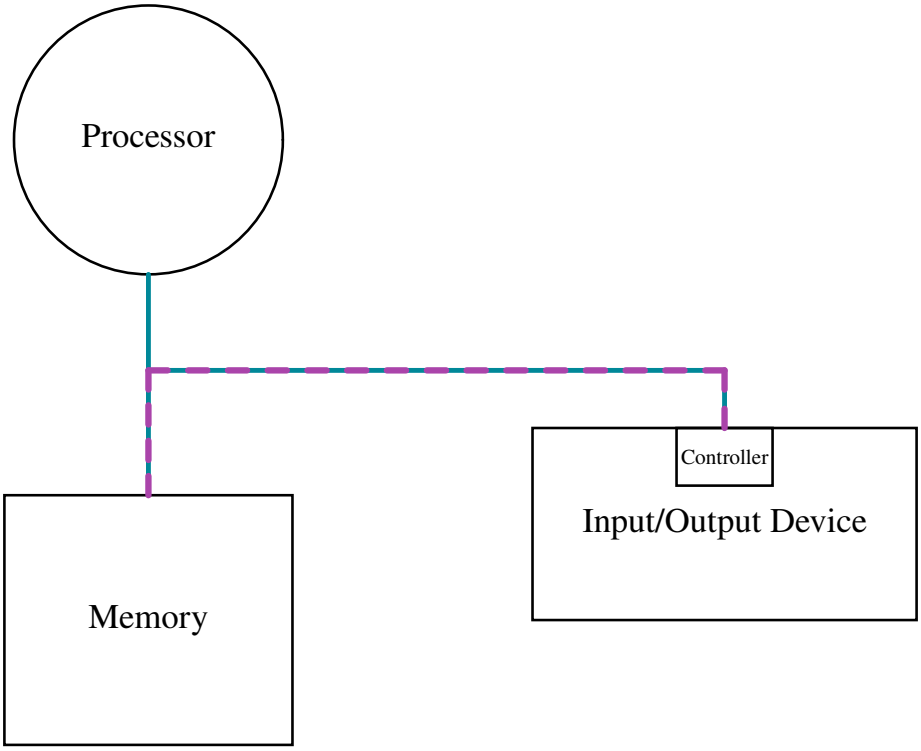
- لازم است اطلاعات دستگاه‌های ورودی یا خروجی به حافظه انتقال یابد.
- در این شرایط پردازنده به عنوان واسط عمل می‌کند.



- این کار چه بدی دارد؟

- ویژگی Direct Memory Access (DMA):

دستگاه ورودی و خروجی به صورت موقتی مستقیما به حافظه‌ی اصلی دسترسی پیدا می‌کند.



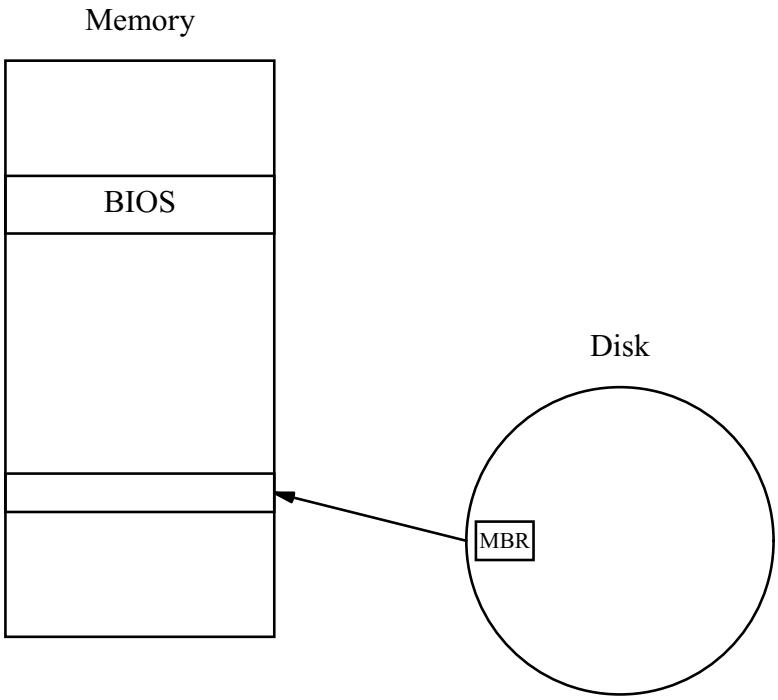
- مزیت این کار چیست؟

- پس از روشن شدن کامپیوتر، گام‌هایی انجام می‌شوند تا سیستم عامل اجرا شود.
- فرض کنید سیستم عامل روی دیسک باشد.
- در بسیاری از سیستم‌های عامل امروزی قبل از سیستم عامل برنامه‌ای به نام Bootloader اجرا می‌شود و این برنامه سیستم عامل را به حافظه انتقال می‌دهد و اجرا می‌کند.
- برای سادگی فرض کنید Bootloader وجود ندارد.

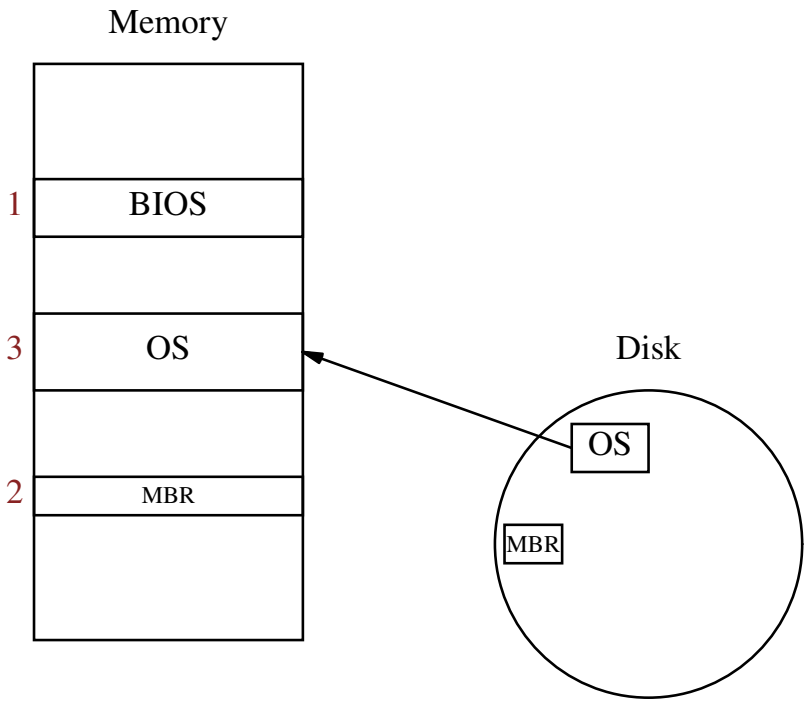
- برنامه‌ای به نام Bootstrap در تراشه‌ای روی Motherboard قرار دارد.

  در معماری x86 نام این برنامه BIOS است.
- وقتی کامپیوتر روشن می‌شود، این برنامه اجرا می‌شود.
- برنامه‌ی Bootstrap قطعات سخت‌افزاری را مقدار دهی اولیه می‌کند و امکان استفاده از آنها را فراهم می‌کند.
- سپس اولین سکتور را (معمولا ۵۱۲ بایت) از دیسک مناسب به حافظه انتقال می‌دهد.

  به این قسمت MBR (Master Boot Record) گفته می‌شود.
- سپس با یک دستور پرش، اجرای آن را شروع می‌کند.



- بخش MBR برای نگهداری کل سیستم عامل کوچک است.
- این قسمت معمولا ادامه‌ی سیستم عامل را از دیسک به حافظه انتقال می‌دهد.



- سپس اجرای سیستم عامل شروع می‌شود.
- سیستم عامل سخت‌افزار را مقداردهی اولیه می‌کند تا آماده‌ی استفاده شود.
- سپس پردازه‌های کاربری اجرا می‌شوند.
- در واقعیت معمولا ابتدا برنامه‌ای به نام Bootloader به حافظه انتقال می‌یابد و اجرا می‌شود.

سپس این برنامه سیستم عامل را به حافظه انتقال می‌دهد و اجرای سیستم عامل شروع می‌شود.

- سیستم عامل سخت‌افزار را مقداردهی اولیه می‌کند.
- سپس اولین پردازه را اجرا می‌کند.