Operating Systems
سیستمهای عامل

اسلایدهای شماره ۳

دکتر خانمیرزا

h.khanmirza@kntu.ac.ir

دانشکده کامپیوتر

دانشگاه صنعتی خواجه نصیرالدین طوسی

مفهوم چهارم: اجرای دو حالته (Dual Mode)

- پردازنده‌های فعلی دارای دو حالت اجرایی هستند
  - حالت هسته
    - یا حالت کرنل - حالت Supervisor - حالت محافظت شده)
      - سیستم عامل در این حالت اجرا میشود
  - حالت کاربر: حالت عادی اجرای برنامه
- حالت هسته
  - پردازنده در حالت کرنل دستورات با تقدم (privileged) را اجرا می‌کند.
  - برخی دستورات در پردازنده هستند که فقط در حالت هسته قابل اجراست.
  - حالت اجرا بوسیله یک بیت در پردازنده مشخص می‌شود که الان باید پردازنده در چه حالتی کار کند
  - تغییر این بیت خودش دستور باتقدم است و فقط توسط سیستم عامل قابل انجام است.

مفهوم چهارم: اجرای دو حالته (Dual Mode)

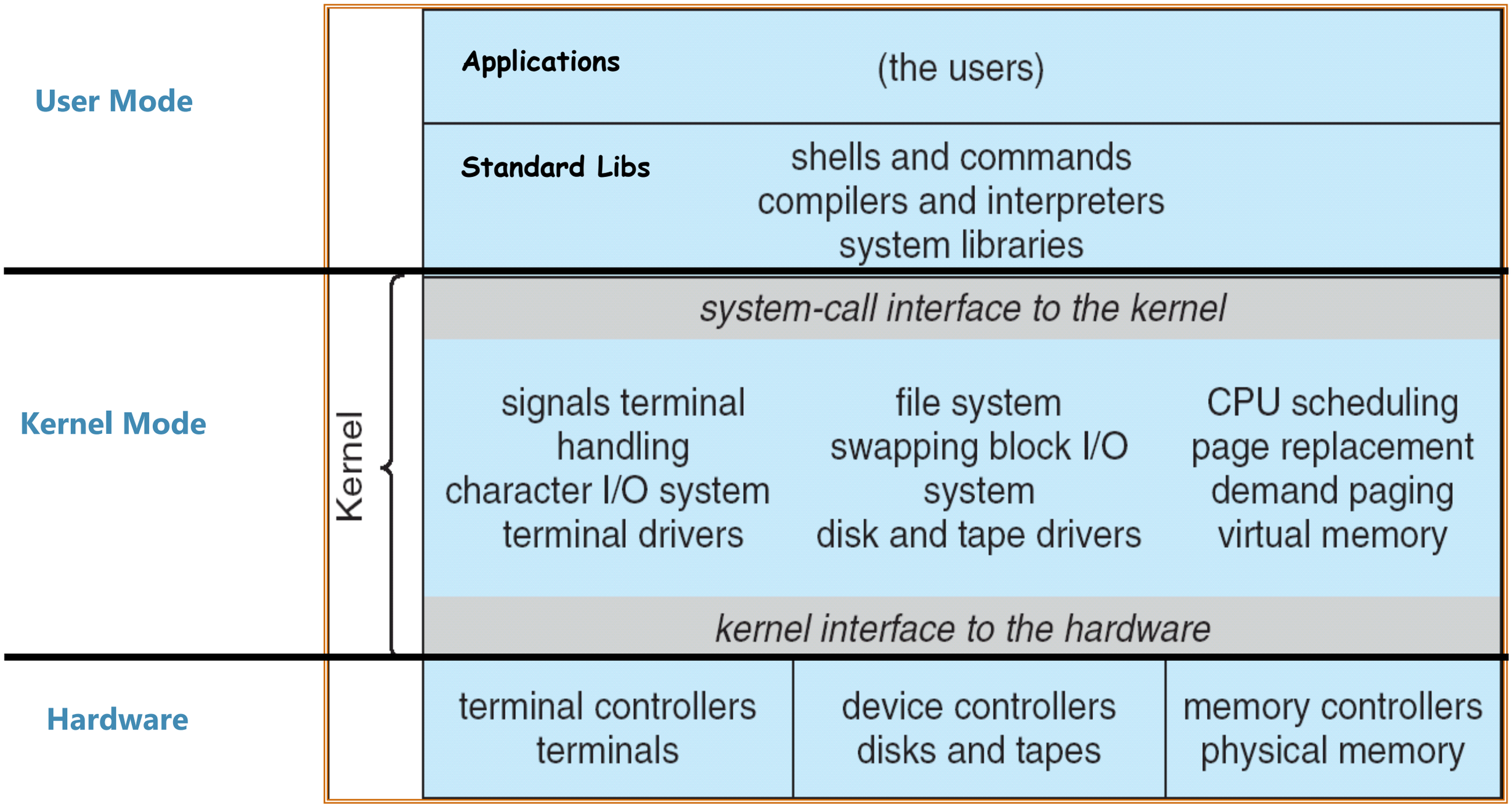
- چه دستوراتی **با تقدم بالا** هستند؟
  - دستورات مدیریت حافظه
  - تغییر زمینه اجرا (= تغییر ریسمان اجرایی)
  - دسترسی به تجهیزات سخت‌افزاری جانبی
  - ....
- اگر ریسمانی در حالت عادی کار می‌کند بخواهد دستور با تقدمی اجرا کند دچار خطا می‌شود و اجرای برنامه قطع خواهد شد.
- طبیعی است که تغییر به حالت اجرایی هسته باید سخت باشد تا هر ریسمانی نتواند براحتی اجرای خود را به این حالت اجرایی تغییر دهد

مفهوم چهارم: اجرای دو حالته (Dual Mode)

User Mode

Kernel Mode

Hardware



- معماری سیستم عامل unix

مفهوم چهارم: اجرای دو حالته (Dual Mode)

- تنها ریسمانی که در حالت هسته اجرا می‌شود سیستم عامل است و از آنجا که بقیه ریسمانها و فرآیندها توسط سیستم عامل ایجاد و اجرا می‌شوند (بعدا صحبت می‌شود) بنابراین هیچ ریسمان دیگری نمی‌تواند در حالت هسته باشد
  - دقت کنید که تغییر بیت اجرایی پردازنده خودش دستور با تقدم است
- سیستم عامل چطور می‌تواند این توانایی را داشته باشد. لازم است که به فرآیند Boot توجه کنیم

فرآیند بوت

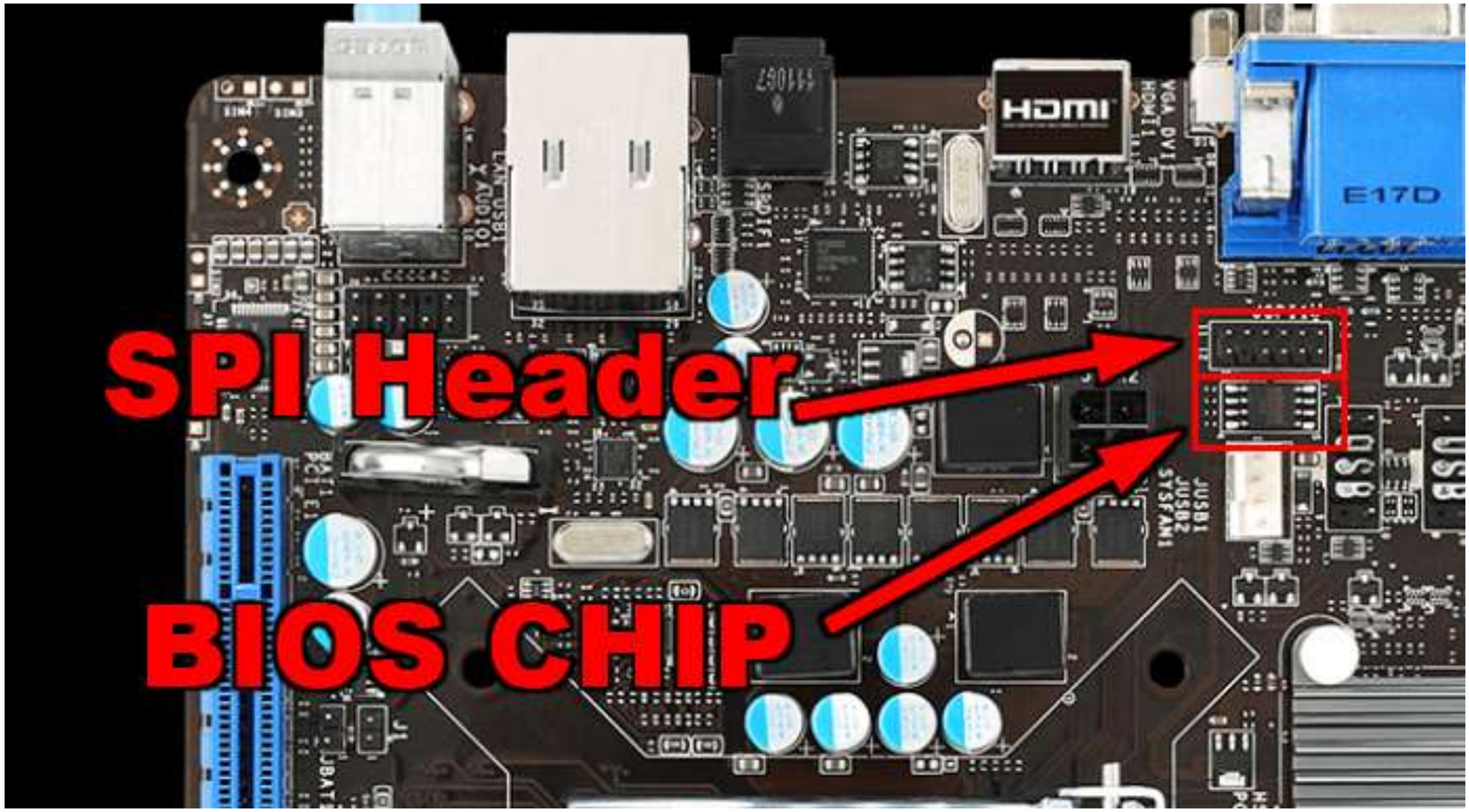
- بوت (Boot):
  - فرآیندی که از زمان فشردن دکمه روشن کردن سیستم (power) تا بالا آمدن سیستم عامل انجام می‌شود بوت می‌گوییم

- بردهای مادر (Motherborad) دارای یک تراشه ROM و یا در سیستمهای جدید E^2PROM هستند که در داخل آن کدی به نام BIOS (Basic Input/Output System) ذخیره می‌شود.
  - کدهایی که برای کنترل و کاربری تجهیزات سخت‌افزاری در سطح پایین نوشته می‌شوند و عموما در ROM ذخیره می‌شوند firmware گفته می‌شود

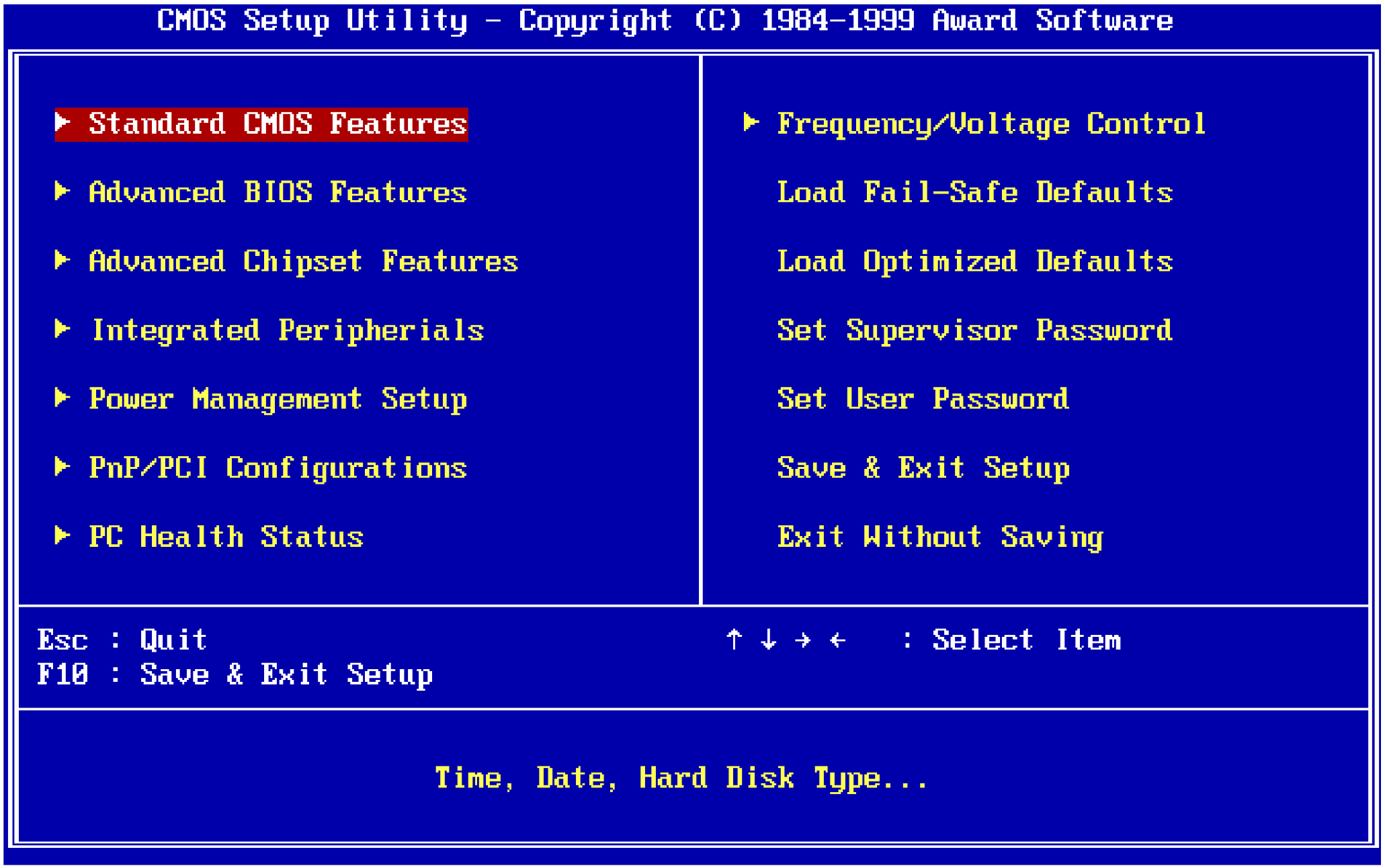
- BIOS یک firmware است که دارای کدهای سطح پایین برای شروع اولیه سیستم است

- در این کد سخت‌افزارهای اولیه و ضروری برای شروع به کار سیستم (نظیر ماوس، کیبورد، هارد دیسک و ..) تست و بعد راه‌اندازی می‌شوند.
  - برخی BIOSها دارای یک صفحه برای نمایش پیشرفت بوت هستند
  - در حال حاضر بیشتر BIOSها یک برنامه کامل برای تغییر تنظیمات سیستم هستند

فرآیند بوت



فرآیند بوت


CMOS Setup Utility - Copyright (C) 1984-1999 Award Software
Standard CMOS Features
Advanced BIOS Features
Advanced Chipset Features
Integrated Peripherials
Power Management Setup
PnP/PCI Configurations
PC Health Status
Frequency/Voltage Control
Load Fail-Safe Defaults
Load Optimized Defaults
Set Supervisor Password
Set User Password
Save & Exit Setup
Exit Without Saving
Esc : Quit
F10 : Save & Exit Setup
↑ ↓ → ← : Select Item
Time, Date, Hard Disk Type...

فرآیند بوت

- پس از راه‌اندازی سخت افزارها، برنامه BootLoader فراخوانی و مدیریت سیستم در اختیار این برنامه قرار می‌گیرد
  - مانند برنامه GRUB در لینوکس
  - این برنامه‌ها اختیار انتخاب به کاربر می‌دهند
    - اگر سیستم دارای چندین سیستم عامل باشد کاربر یکی را انتخاب و با آن کار می‌کند
    - می‌توانند برخی پارامترهای سیستم عامل را تغییر دهند
- پس از آن برنامه BootLoader به ابتدای آدرس سیستم عامل ذخیره شده پریده و اجرای دستورات از آنجا ادامه می یابد. در این مرحله سیستم عامل لود شده و اجرا می‌شود.
- در این مرحله سیستم عامل اجرای هسته را برای خود فعال کرده و همیشه در این حالت به اجرای خود ادامه می‌دهد.

روشهای تغییر حالت اجرایی

- فرض کنید که یک ریسمان می‌خواهد در دیسک مقداری را بنویسد.
- چون دیسک سخت‌افزار است این کار فقط در حالت باتقدم قابل انجام است و کنترل آن در دست سیستم عامل است این ریسمان چطور کار خود را باید انجام دهد؟

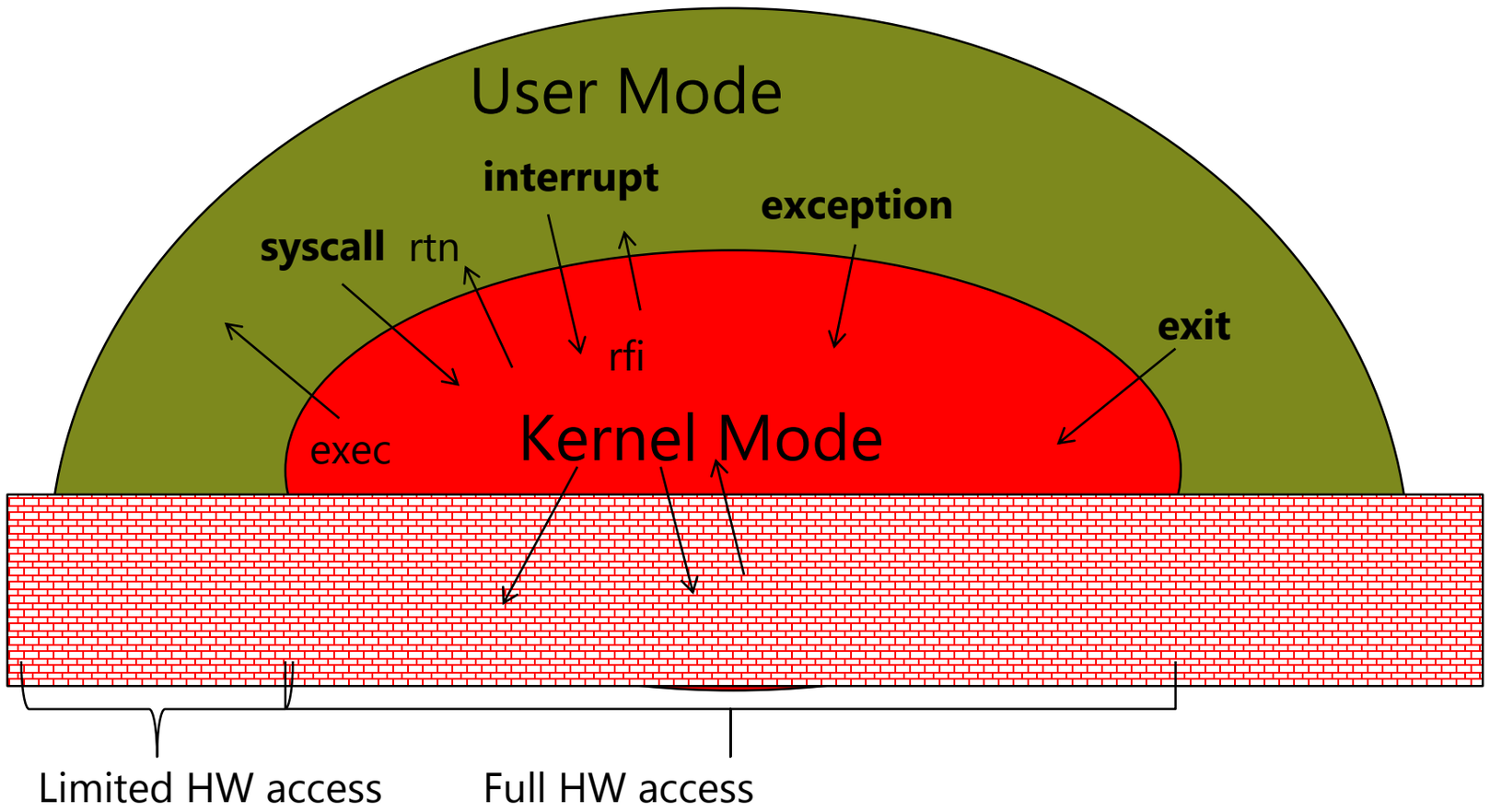
- برای اجرای کارهای باتقدم فرآیند باید به سیستم عامل درخواست بدهد
- پس از آن تمامی اطلاعات لازم از حافظه ریسمان به حافظه هسته کپی شده و کد در سیستم عامل اجرا می‌شود.
- پس از اجرا نتیجه به حافظه کاربری منتقل می‌شود

- برای چنین کاری سه روش انتقال ریسمان به حالت هسته وجود دارد

روشهای تغییر حالت اجرایی

- این کار مانند انجام دادن کارهای بانکی در شعبه به صورت حضوری است
  - شما دسترسیهای لازم برای انجام کار بانکی مد نظر را ندارید
  - پس از مراجعه به شعبه بانک تقتضای خود را با کارمند بانک مطرح می کنید
  - کارمند بانک دسترسیهای لازم را دارد و بادریافت برخی اطلاعات، از طرف شما کار را انجام می دهد
  - پس از انجام نتیجه را با یک برگه چاپ شده به شما اطلاع می دهد

مفهوم چهارم: اجرای دو حالته (Dual Mode)



روشهای تغییر حالت اجرایی

- سه روش برای انتقال حالت کاری وجود دارد:
- فراخوانی سیستمی (system-call):
  - زمانی که یک سرویس سیستمی از سیستم عامل می‌خواهیم
  - این کار شبیه فراخوانی تابع در برنامه معمولی است منتها آدرس تابع را نمیدانیم. فقط نام سرویس و آرگومانها را به سیستم عامل می‌دهیم و سیستم عامل از طرف ریسمان کار را انجام می‌دهد

- وقفه (interrupt):
  - یک رویداد ناهمگام که بیرون از فرآیند رخ داده است و وابسته به فرآیند نیست مثل آمدن یک بسته شبکه، فشرده شدن دکمه صفحه کلید، تایمر زمان اجرای برنامه و ...
  - این رویدادها زمانی به فرآیند اطلاع داده میشوند که درخواست دریافت آن به اطلاع سیستم عامل رسیده باشد.

- خطا (trap یا exception) :
  - یک رویداد همگام در داخل فرآیند که باعث تعویض زمینه میشود

- در هر سه نوع انتقال حالت برنامه‌ریزی نشده است یعنی ریسمان نمی‌داند تابع مربوط به سرویسها کجا بوده و قرار است کجا اجرا شود

انتقال امن در تغییر حالت اجرایی

- فرآیند انتقال حالت اجرایی از کاربری به هسته باید به صورت امن انجام شود
  - در غیر این صورت، وجود دو حالت اجرایی بی معنی میشود

- مکانیزمهای انتقال امن حالتهای اجرایی
  - انتقال کنترل شده اطلاعات (Controlled transfer)
  - پشته دوگانه (Dual Stack)

انتقال امن در تغییر حالت اجرایی – انتقال کنترل شده

- انتقال کنترل شده
  - یک کد که به دقت نوشته شده وضعیت فعلی ریسمان را ذخیره می‌کند
  - آرگومانهای لازم برای انتقال به حالت هسته در ثباتها ذخیره می‌شوند
  - آرگومانها در حافظه سیستم عامل کپی می‌شوند (کجا؟)
  - قبل از انتقال آرگومانها و مقادیر با جدیت بررسی میشوند تا کد و داده مخرب در سیستم عامل کپی نشود
    - مثلا
      - `fwrite(void * buffer, size_t size, size_t count, FILE * stream)`
      - `fwrite(buffer, sizeof(struct my_data, -10, file);`
    - این کار در توابع مختلفی نظیر memcpy و ... هم مهم است.
  - کد مورد نظر در حالت با تقدم اجرا شده و نتیجه در ثباتهای پردازنده ذخیره می‌شود
  - نتیجه به پشته کاربری (که در حافظه کاربری است) کپی می‌شود

انتقال امن در تغییر حالت اجرایی – انتقال کنترل شده

- سوال: چرا با وجود اینکه سیستم عامل به تمامی حافظه دسترسی دارد آرگومانها باید به هسته کپی شوند.
- چرا مستقیما از پشته کاربری استفاده نمی‌شود؟

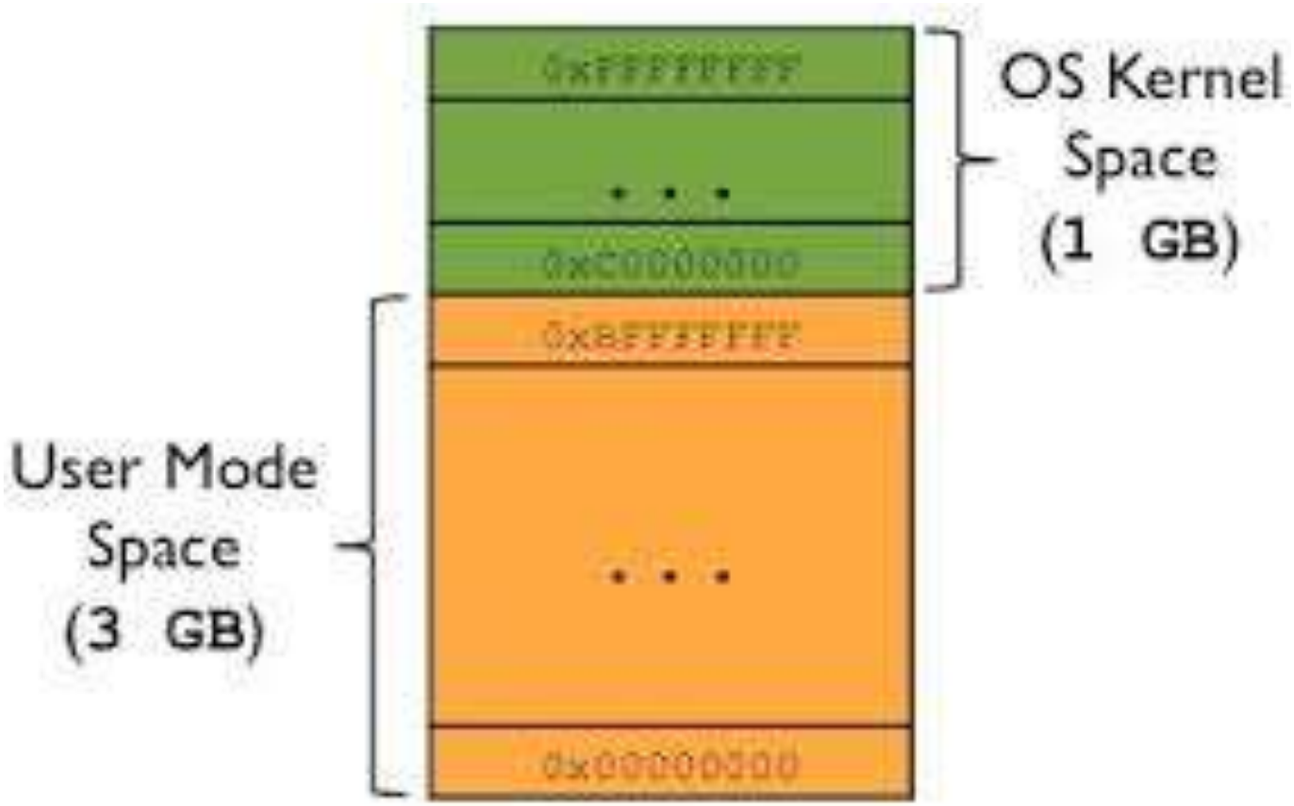
- بدلیل امنیت
  - اگر سیستم عامل کارهای با تقدم را در حافظه کاربری انجام دهد سایر ریسمانها و فرآیندها می‌توانند به اطلاعات حیاتی سیستم عامل دسترسی پیدا کنند
  - در حالت کاربری ضمانتی وجود ندارد که مقادیر آرگومانها دستکاری نشود
    - سیستم عامل مانند سایر فرآیندها به صورت همزمانی اجرا می‌شود، بنابراین برخی زمانها در پردازنده نیست و فرآیندهای دیگر در پردازنده در حال اجرا هستند
    - ریسمانهای یک فرآیند به شکل معمول و ریسمانهای فرآیندهای دیگر با روشهای غیرمعمول مثل hacking می‌توانند حافظه ریسمان را دستکاری کرده و عملکرد سیستم عامل را تخریب کنند

- چرا با کپی مقادیر در حافظه سیستم عامل امنیت برقرار می‌شود؟

انتقال امن در تغییر حالت اجرایی – انتقال کنترل شده

- سیستم عامل حافظه سیستم را به دو قسمت تقسیم می‌کند
  - یک بخش (معمولا یک چهارم) از حافظه را به خود اختصاص میدهد.
  - بقیه فضای حافظه در اختیار فرآیندهای کاربری قرار می‌گیرد

- نحوه اختصاص فضای آدرسها و ترجمه آن به گونه‌ای است که فرآیندها به هیچ وجه به حافظه سیستم عامل دسترسی ندارند و حتی نمی‌توانند آدرسهای این ناحیه را تولید کنند



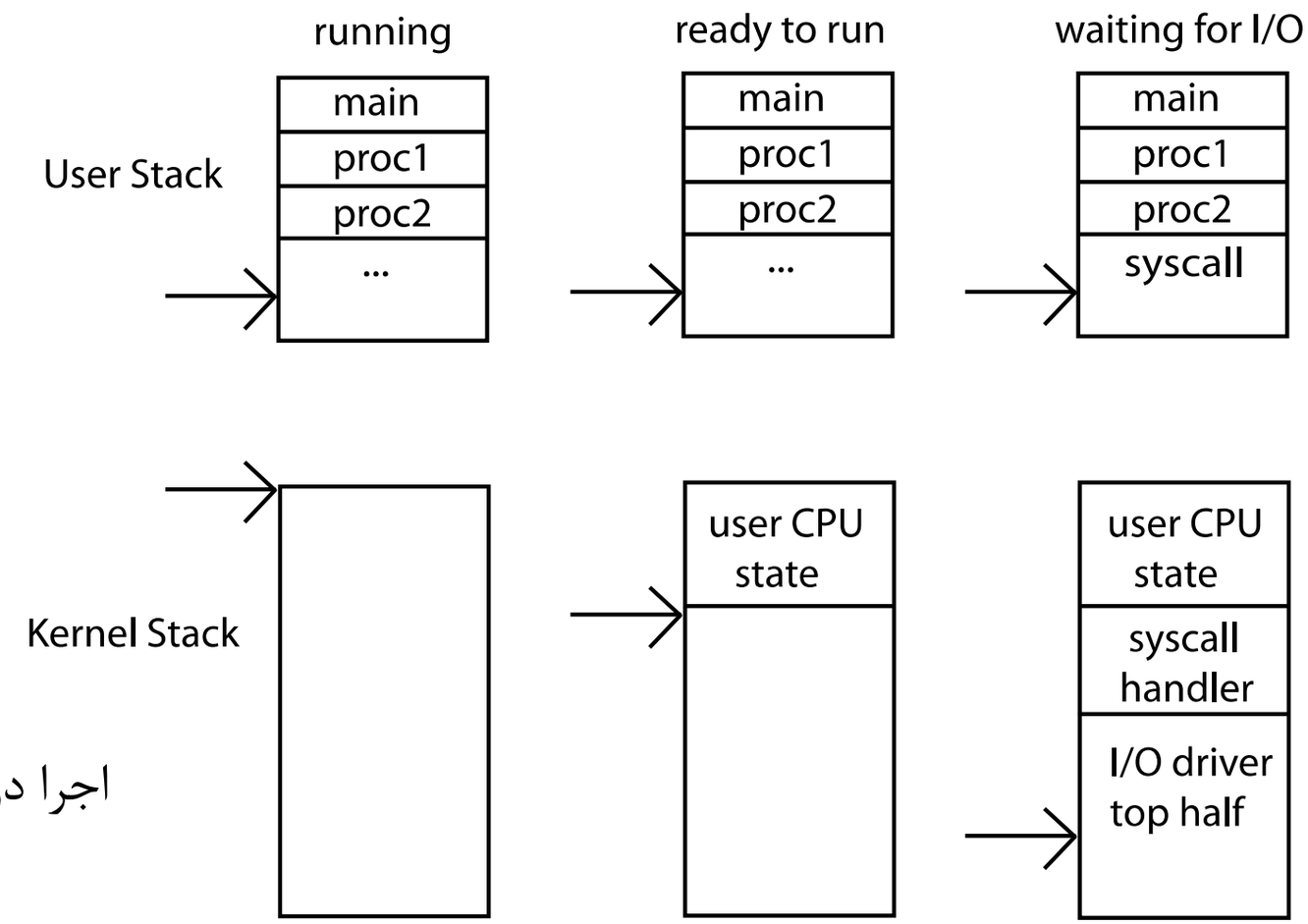
انتقال امن در تغییر حالت اجرایی – پشته دوگانه

- سیستم عامل تابع مرتبط با فراخوانی سیستمی را چطور اجرا می‌کند؟
- آرگومانهایی که از پشته کاربری در حافظه سیستم عامل کپی می‌شوند کجا کپی می‌شوند؟

- اجرای برنامه‌ها طبق استاندارد پردازنده‌ها نیاز به پشته دارد.
  - بدلایل ذکر شده نمی‌توان از پشته فرآیند در حالت کاربری استفاده کرد

- در بیشتر سیستم عاملهای امروزی هر ریسمان دارای دو پشته است یکی در حافظه کاربری و دیگری در حافظه سیستم عامل
  - در گذشته سیستم عامل‌ها کلا یک پشته و یا برای هر هسته یک پشته داشتند.
  - پشته دوم یک بخش حافظه سیستم عامل است که برای هر ریسمان اختصاص داده می‌شود

انتقال امن در تغییر حالت اجرایی – پشته دوگانه



اجرا در حالت کاربری

اجرای سرویس درخواستی ریسمان در حالت هسته و در حال انتظار برای دریافت یک وقفه

اجرای سرویس درخواستی ریسمان در حالت هسته

روشهای تغییر حالت اجرایی

- فراخوانی سیستمی (System Call)
  - زمانی که قرار است یک ریسمان یک کار با تقدم انجام دهد آنرا با فراخوانی سیستمی از سیستم عامل می‌خواهد.
  - این درخواست با فراخوانی یکی از توابع معرفی شده در API سیستم عامل انجام می‌گیرد
  - معمول‌ترین فراخوانی‌های سیستمی ارتباط با سخت‌افزار است
    - مثل خواندن و نوشتن در فایل که در حقیقت ارتباط با دیسک است (fread, fwrite)

- در ظاهر فراخوانی سیستمی تفاوتی با فراخوانی عادی توابع ندارد اما آنچه در پس این فراخوانی وجود دارد کاملا متفاوت است.

روشهای تغییر حالت اجرایی

- مراحل فراخوانی سیستمی (System Call)

1) نام تابع سیستمی بر اساس یک جدول به یک شماره تبدیل می‌شود
2) این شماره به سطر یک جدول اشاره می‌کند که در هر سطر این جدول آدرس تابع handler یعنی تابعی که این سرویس را در سیستم عامل پیاده‌سازی کرده است مشخص می‌کند.
3) مقادیر آرگومانها در ثباتهای مشخصی قرار می‌گیرد. برای تعداد آرگومانهای بیشتر و آرگومانهای بزرگ مثل آرایه‌ها، اشاره‌گر آنها در ثباتها قرار گرفته و مقادیر مستقیما از پشته ریسمان به پشته هسته کپی می‌شود.
4) در اینجا حالت اجرایی پردازنده به حالت هسته تغییر می‌کند
5) آرگومانها و مقادیر آنها به شکل جدی بررسی می‌شوند تا از جهت اجرایی امن باشند
6) تابع سیستمی با کمک پشته هسته ریسمان اجرا می‌شود
7) نتیجه نهایی در ثباتها قرار می‌گیرد و یا ممکن است به پشته کاربر مستقیما کپی شود
8) حالا فراخوانی سیستمی تمام شده و return انجام می‌گیرد

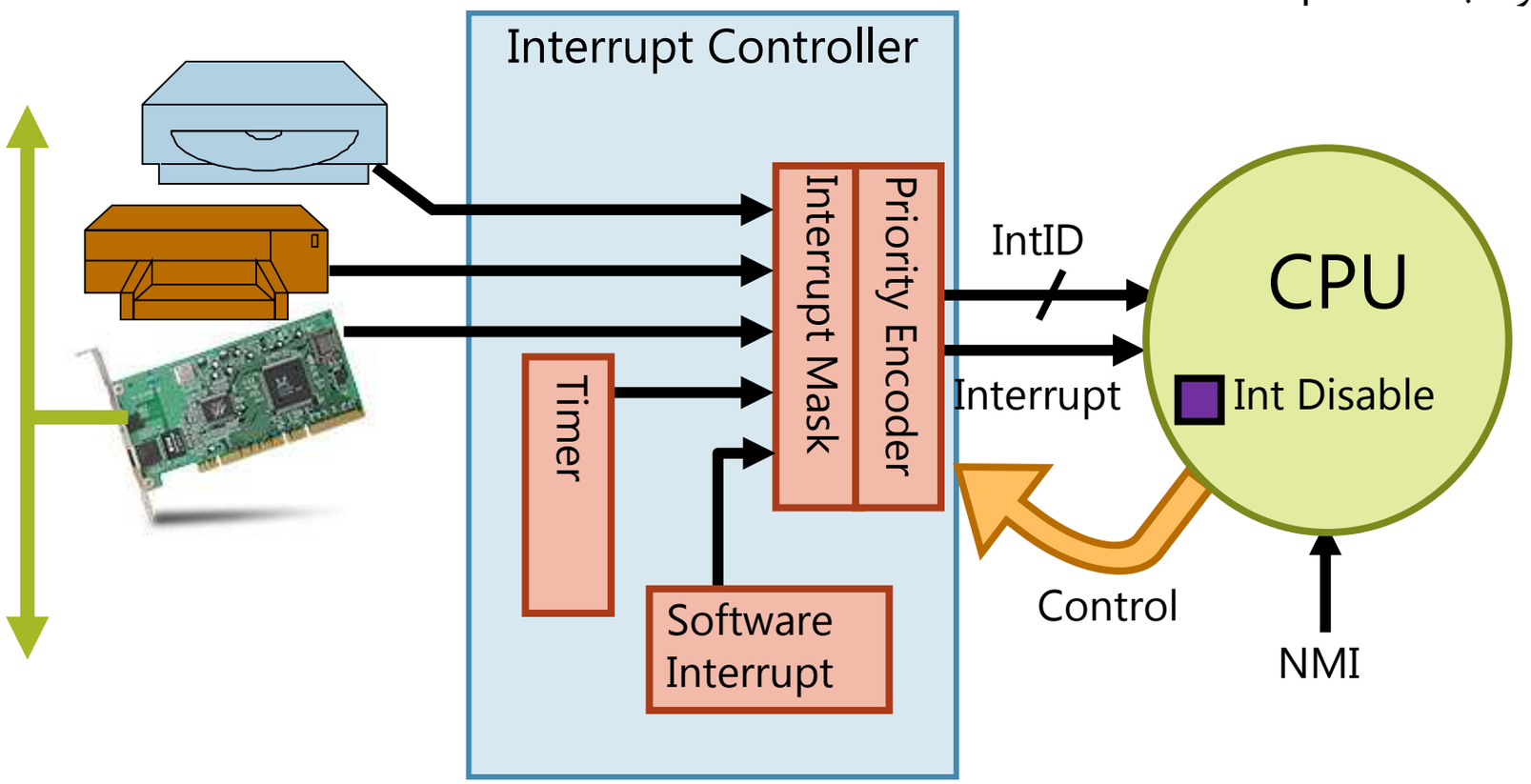
روشهای تغییر حالت اجرایی

- وقفه (interrupt)
  - وقفه‌ها عموما یک رویداد ناهمگام (asynchronous) است و زمان وقوع آن مشخص نیست
    - وقفه معمولا در وسط اجرای یک ریسمان دیگر اتفاق میفتد
  - فراخوانی سیستمی یک رویداد همگام (synchronous) است
    - فراخوانی سیستمی توسط خود برنامه‌نویس در کد و در محل مشخص فراخوانی می‌شود

- وقفه‌ها به شکل اسرارآمیزی کار می‌کنند!
- نه کابران و نه ریسمانهای در حال اجرا از زمان و نحوه اجرای وقفه‌ها بی‌اطلاع هستند

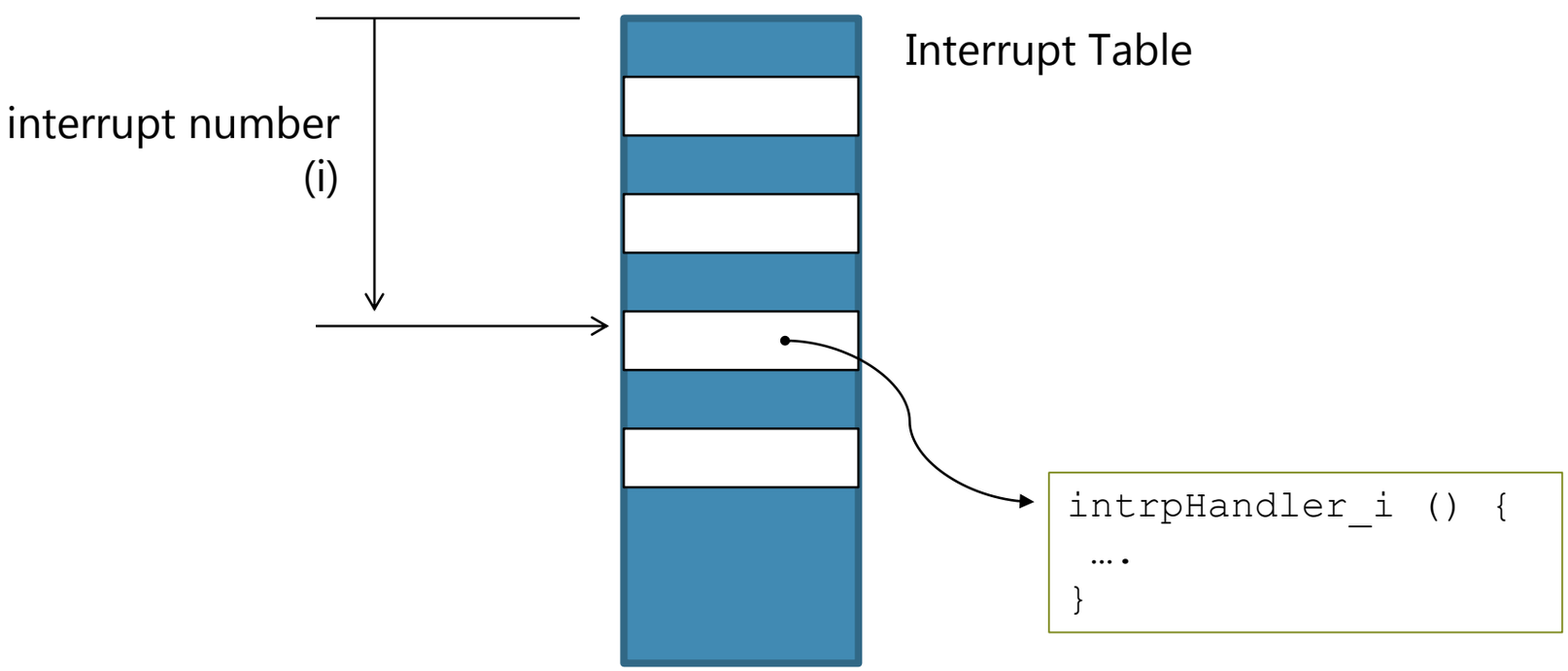
روشهای تغییر حالت اجرایی

- وقفه (interrupt)
  - همه سیستمها دارای یک تراشه کنترل وقفه هستند که تمامی تجهیزات جانبی (Peripherals) به این تراشه متصل هستند
  - زمانی که هر کدام از تجهیزات جانبی با پردازنده کار داشتند سیم مربوط به خود را فعال می‌کنند بدین معنی که با پردازنده کار دارند.
  - درخواستها بر اساس تقدم اولویت داده می‌شوند و شماره تجهیزی که اولویت دارد به پردازنده اعلام می‌شود
  - پردازنده می‌تواند با یک flag داخلی همه وقفه‌ها را غیرفعال کند و یا با برنامه‌ریزی ثبات Interrupt Mask فقط برخی وقفه‌ها را فعال نماید
  - برخی وقفه‌ها non-maskable هستند و عمدتا مواردی هستند که سیستم عامل نمی‌تواند در قبال آنها کاری انجام دهد
    - خطای حافظه به نحوی که قابل بازیابی نیست
    - خطای مربوط به power



روشهای تغییر حالت اجرایی

- وقفه (interrupt)
  - زمانی که پردازنده از وجود وقفه مطلع می‌شود وضعیت ریسمان فعلی را ذخیره و از پردازنده خارج می‌کند
  - سپس شماره تجهیزی که درخواست دارد به پردازنده اعلام می‌شود
  - شماره تجهیز به سطر بردار وقفه (interrupt vector) اشاره می‌کند که در آن آدرس تابع handler وقفه مشخص شده است
  - پردازنده اجرای دستورات را از محل تابع مشخص شده ادامه می‌دهد
    - اجرای وقفه با کمک پشته وقفه انجام می‌گیرد که گاهی برای هر هسته یک پشته وقفه در نظر گرفته می‌شود
  - پس از اتمام تابع وقفه، پردازنده به اجرای ریسمان قبلی باز می‌گردد



روشهای تغییر حالت اجرایی

- در هر لحظه هر پردازنده در یکی از وضعیتهای زیر است
  - در حال اجرای یک ریسمان در حالت کاربری
  - در حال اجرای یک ریسمان در حالت هسته
  - در حال اجرای یک وقفه در حالت هسته

روشهای تغییر حالت اجرایی

- اگر وقفه وجود نداشت باید از سرکشی (polling) استفاده می‌کردیم
  - باید سخت‌افزارها به شکل پریودیک بررسی می‌شدند تا ببینیم کدامیک درخواست دارند.

- وقفه چه مزیتی نسبت به سرکشی دارد
  - برخی تجهیزات بندرت با پردازنده کار دارند مثلا زمانی که صدایی پخش نمی‌شود کارت صدا کاری ندارد. سرکشی به آنها باعث اتلاف زمان می‌شود
  - برخی تجهیزات بسیار کند هستند (مانند دیسک) در زمان سرکشی باید پردازنده منتظر پاسخ آنها بماند

- چه زمانی سرکشی به وقفه ارجحیت دارد؟

روشهای تغییر حالت اجرایی

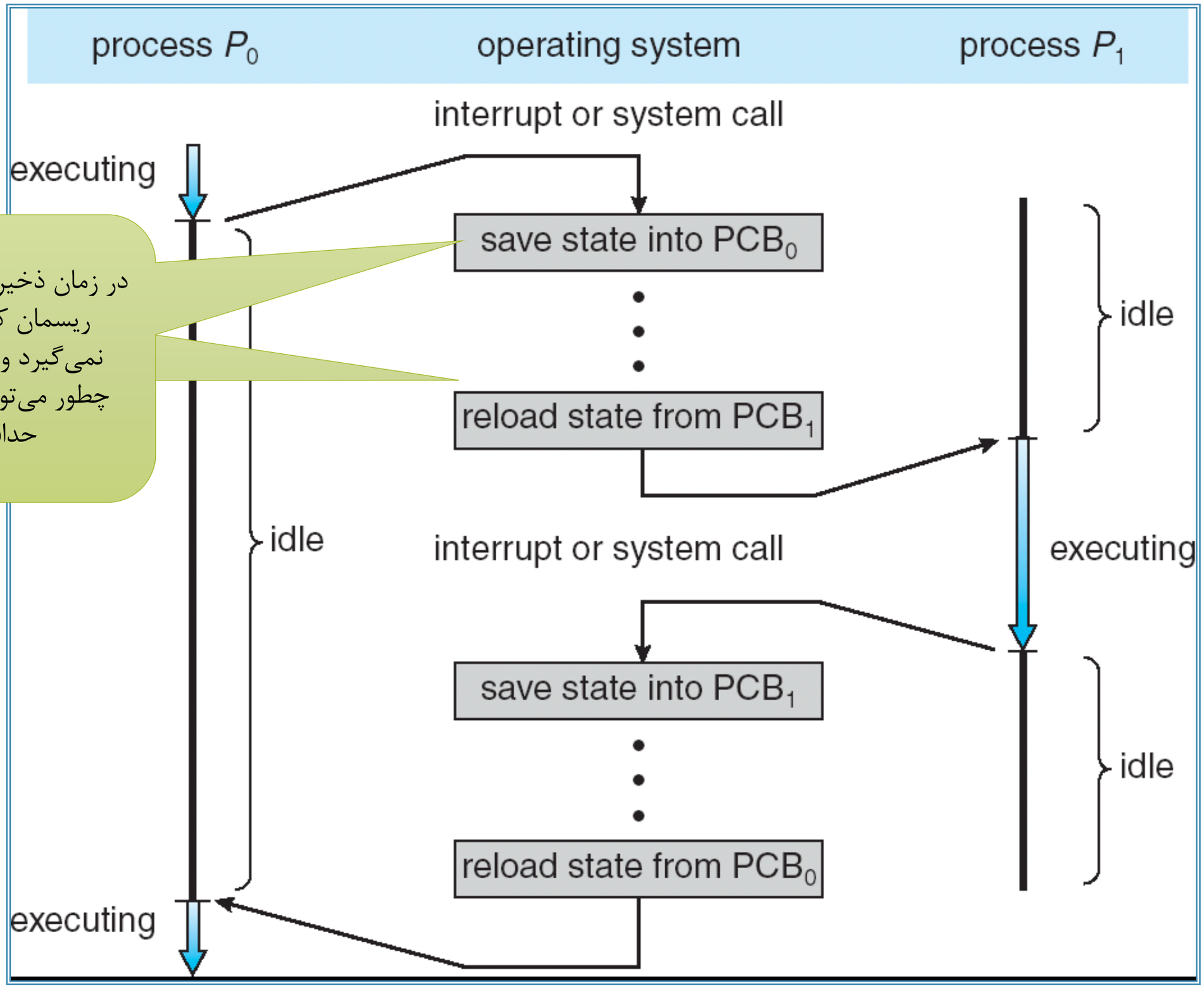
- خطا (trap, exception)
  - یک وقفه همگام است که توسط نرم‌افزار و در شرایط خاص و استثنایی تولید می‌شود.
    - تقسیم بر صفر
    - دسترسی به حافظه اشتباه (یا به جایی که فرآیند دسترسی ندارد)
    - اجرای دستورهای باتقدم در حالت کاربری
    - debugger (اجرای تک به تک دستورات)
      - بین دستورات trap وارد می‌شود که بخاطر آن اجرا متوقف شده و مقادیر وضعیت پردازنده به ریسمان در حال اجرا منتقل شده و منتظر می‌ماند تا کاربر دوباره دکمه‌ای را فشار دهد
  - همانند وقفه، خطا باعث تعویض اجرای حالت کاربری به حالت هسته می‌شود.
    - روال اجرای trap همانند وقفه و فراخوانی سیستمی است
      - شماره خطا - جدول خطا - tap handler
      - برخی سیستم عاملها فراخوانی سیستمی و خطا را با هم پیاده‌سازی کرده‌اند
    - گاهی مواقع عمداً در کد یک trap فراخوانی می‌شود تا از تابع خطا برای مقاصد درست استفاده کنیم
      - بعدا در بخش مدیریت حافظه خواهیم دید

تعویض زمینه (context switch)

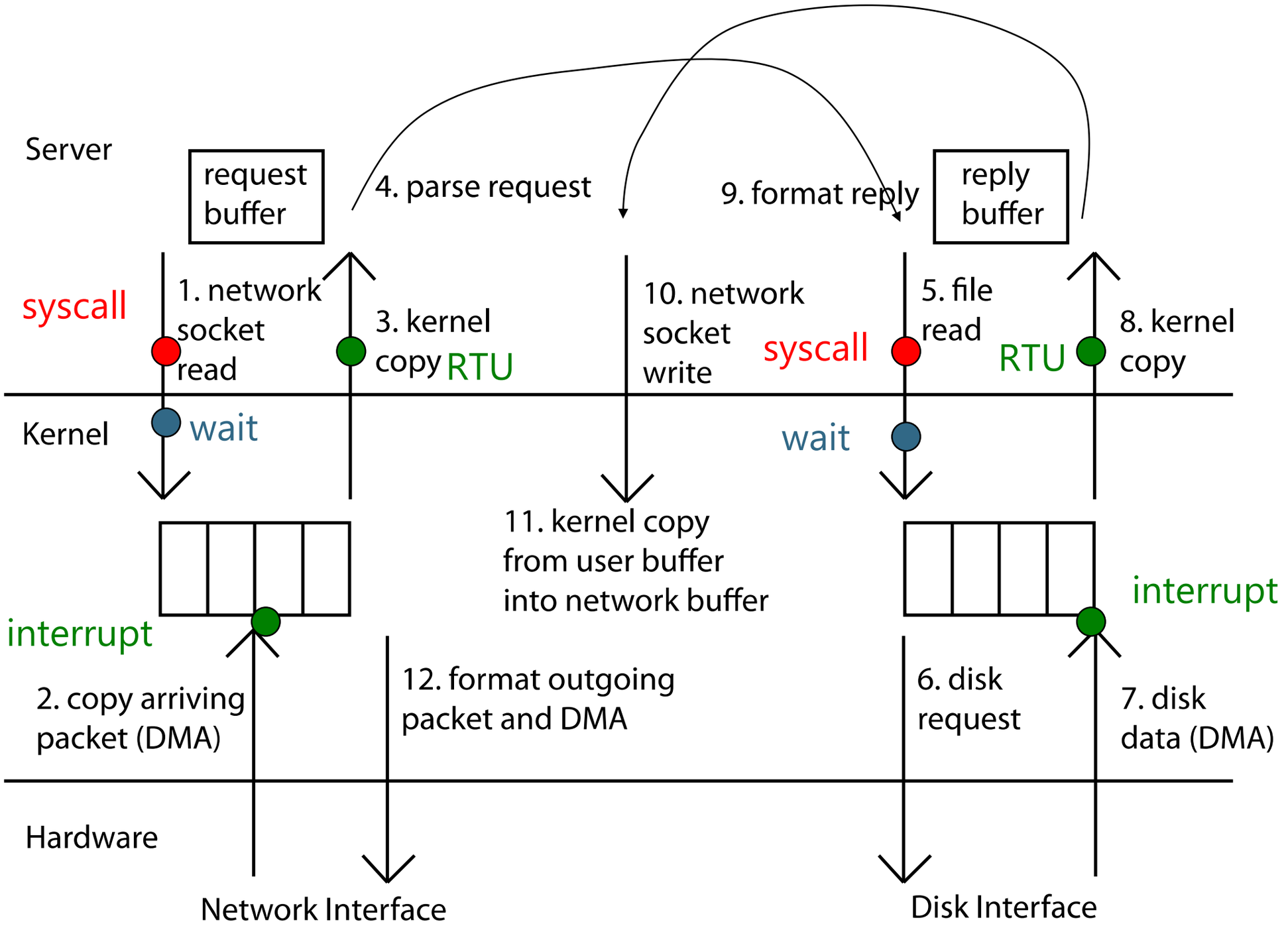
- تعویضِ ریسمانِ در حالِ اجرا و لود کردن یک ریسمان جدید در پردازنده برای اجرا را تعویض زمینه می‌گوییم
- تعویض زمینه در اثر وقفه سخت افزار تایمر اجرا می‌شود
  - همانطور که گفته شد زمانی که یک وقفه رخ می‌دهد ریسمان فعلی از پردازنده خارج و تابع وقفه اجرا می‌شود.
  - تابع handler وقفه‌ی تایمر همان کد تعویض زمینه است
- مکانیزم همزمانی (concurrency) نیز همان اجرای کد تعویض زمینه است
  - تایمر یک سخت افزار در سیستم است
  - زمانی که یک ریسمان در پردازنده برای اجرا قرار می‌گیرد زمانی برای اجرای آن در نظر گرفته می‌شود
  - هر گاه زمان به سر برسد تایمر یک وقفه تولید می‌کند
  - بر اثر وقفه تایمر تابع تعویض زمینه اجرا شده و ریسمان دیگری جایگزین ریسمان فعلی می‌شود.
  - اطلاعات زمینه اجرایی (execution context) ریسمان قبلی در جایی ذخیره می‌شود تا بار بعدی که برای اجرا انتخاب شد اجرای ریسمان از همان جایی که مانده بود ادامه یابد.

تعویض زمینه (context switch)

در زمان ذخیره و بازیابی اطلاعات ریسمان کار مفیدی انجام نمی‌گیرد و همه سربار است. چطور می‌توان این سربار را به حداقل رساند؟



بررسی یک سناریو



بررسی یک سناریو

- یک وب سرور را در نظر بگیرید که میخواهد درخواست بعدی را از کرنل بخواند.
  - ریسمان یک فراخوانی سیستمی برای خواندن بسته از کارت شبکه انجام می‌دهد
  - در بافر کارت شبکه بسته‌ای نیامده است بنابراین فرآیند به حالت انتظار می‌رود
    - وضعیت pcb به waiting تغییر می‌یابد
    - pcb در صف waitingهای آن کارت شبکه قرار داده می‌شود
  - بعد از مدتی یک بسته به کارت شبکه میرسد. این بسته دریافت شده و یک وقفه را شروع می‌کند.
  - ریسمان فعلی متوقف و در صف readyها قرار داده می‌شود و روتین وقفه اجرا می‌شود
  - در اجرای وقفه بسته از بافر کارت به حافظه هسته سیستم عامل کپی می‌شود
    - هر تجهیز در سیستم عامل یک بافر ring دارد که در بخش درایور آن است
  - در انتهای وقفه ریسمانی که منتظر دریافت بسته بود توسط سیستم عامل بیدار می‌شود
    - وضعیت ریسمان به ready تغییر می‌یابد
    - ریسمان به صف فرآیندهای ready منتقل می‌شود
  - بسته از حافظه سیستم عامل به پشته کاربری ریسمان کپی میشود
- همان مراحل را برای نوشتن در دیسک و ارسال پاسخ از کارت شبکه هم وجود دارد.