به نام خدا

دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

درس سیستم های عامل

دکتر زینب زالی

تکلیف اول

پاسخنامه تکلیف

سوال ۱.۱

عبارات و اصطلاحات زیر را توضیح داده و با هم مقایسه کنید

الف)system program و application program

1. System Program (برنامه سیستمی):

برنامه سیستمی معمولاً برنامه‌ای است که به سیستم‌عامل مربوط می‌شود اما لزوماً بخشی از هسته (kernel) نیست. این نوع برنامه‌ها معمولاً به اجرای وظایف پایه‌ای و ضروری سیستم کمک می‌کنند که برای کارکرد صحیح سیستم‌عامل و سخت‌افزار مورد نیاز هستند.

مثال:

Shell در سیستم‌عامل‌های یونیکس و لینوکس یک برنامه سیستمی است.

چه کار می‌کند؟

Shell برنامه‌ای است که به کاربر اجازه می‌دهد دستورات مختلفی را وارد کند و با هسته سیستم‌عامل تعامل داشته باشد. مثلاً وقتی که کاربر دستور **ls** را برای لیست کردن فایل‌ها وارد می‌کند، **Shell** دستور را به هسته سیستم‌عامل ارسال می‌کند و نتیجه به کاربر نمایش داده می‌شود.

2. Application Program (برنامه کاربردی):

برنامه کاربردی برنامه‌ای است که مستقیماً به کاربر خدمات ارائه می‌دهد و برای اهداف خاصی طراحی شده است که خارج از عملیات سیستم هستند. این برنامه‌ها معمولاً به‌طور مستقیم توسط کاربر اجرا می‌شوند.

مثال:

Microsoft Word یک برنامه کاربردی است.

چه کار می‌کند؟

Microsoft Word برنامه‌ای است که برای ویرایش و ایجاد اسناد متنی استفاده می‌شود. وقتی که کاربر یک سند جدید ایجاد یا یک سند موجود را باز می‌کند، این برنامه پردازش‌های مرتبط با متن و فرمت‌گذاری آن را انجام می‌دهد، بدون اینکه مستقیماً به عملیات‌های سیستم‌عامل مربوط باشد.

مقایسه:

نقش در سیستم:

برنامه‌های سیستمی به‌طور کلی برای عملکرد صحیح سیستم و مدیریت منابع کامپیوتر طراحی شده‌اند. بدون این برنامه‌ها، تعامل با هسته سیستم و مدیریت منابع سیستم‌عامل ممکن نیست. در مقابل، برنامه‌های کاربردی به کاربران خدمات مستقیم ارائه می‌دهند و به خود سیستم‌عامل مربوط نیستند.

کاربران هدف:
برنامه‌های سیستمی بیشتر برای توسعه‌دهندگان یا مدیران سیستم که با لایه‌های زیرین سیستم‌عامل کار می‌کنند، مفید هستند. از سوی دیگر، برنامه‌های کاربردی برای کاربران نهایی طراحی شده‌اند که به دنبال انجام وظایف روزمره مانند نوشتن، ویرایش و یا پخش محتوای چندرسانه‌ای هستند.

مثال‌ها:
Shell یک مثال از برنامه سیستمی است که تعامل با سیستم‌عامل را ممکن می‌سازد. **Microsoft Word** یک برنامه کاربردی است که به کاربران کمک می‌کند تا اسناد متنی ایجاد و ویرایش کنند.

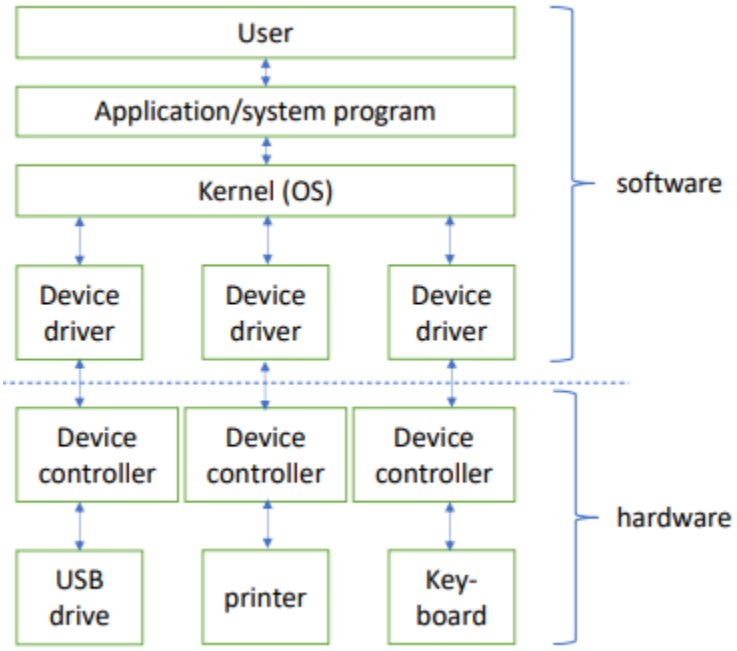
در نهایت، هر دو نوع برنامه برای استفاده از کامپیوتر ضروری هستند، اما یکی به مدیریت و کنترل سیستم کمک می‌کند، در حالی که دیگری به کاربران نهایی برای انجام وظایف خاص خود خدمات می‌دهد.

ب) **device driver** و **device controller** (چگونگی قرار گیری و اتصال آنها در سیستم را به صورت دقیق مشخص کرده و شکل آنها را بکشید و تحلیل کنید.)

Device Driver (درایور دستگاه):
درایور نرم‌افزاری است که در سطح سیستم‌عامل عمل می‌کند و بین سیستم‌عامل و سخت‌افزار واسطه قرار می‌گیرد. وظیفه آن تبدیل دستورات سیستم‌عامل به دستورالعمل‌های قابل فهم برای دستگاه و برعکس است. به عنوان مثال، وقتی سیستم‌عامل می‌خواهد به یک دیسک دسترسی پیدا کند، درایور دستورالعمل‌ها را به زبان کنترل‌کننده دستگاه ترجمه می‌کند.
Device Controller (کنترل‌کننده دستگاه):
کنترل‌کننده بخشی از سخت‌افزار است که به دستگاه‌های فیزیکی مانند دیسک‌ها، چاپگرها و غیره متصل است. کنترل‌کننده وظیفه دارد دستورات دریافت شده از درایور را به دستگاه فیزیکی ارسال کند و همچنین اطلاعات را از دستگاه به سیستم‌عامل بازگرداند. کنترل‌کننده‌ها معمولاً به عنوان واسطی بین سیستم‌عامل و دستگاه عمل می‌کنند.

Device driver	Device controller
نرم افزاری	سخت افزاری
رابط بین سیستم عامل و Device controller	رابط بین دستگاه و CPU و Memory

تبدیل دستورات os به دستورات سطح پایین	ارسال سیگنال برای کنترل دستگاه مربوطه



ج) user mode و kernel mode (مقایسه کنید. لزوم استفاده از این دو mode چیست؟)

برنامه‌هایی که در سطح کاربر اجرا می شوند نسبت به برنامه‌های سطح کرنل دارای دسترسی های محدود تر و کمتری هستند و اجازه دسترسی مستقیم به منابع سیستم را ندارند. به عنوان مثال، اگر برنامه ای در حالت کاربر بخواهد به منابع سیستم دسترسی پیدا کند، ابتدا باید با استفاده از syscalls به دسترسی سطح کرنل دست پیدا کند .

در kernel mode ، برنامه دسترسی مستقیم و بدون محدودیت به منابع سیستم دارد.

لزوم استفاده :

امنیت

با جدا کردن این دو حالت، سیستم‌عامل از دسترسی مستقیم برنامه‌های سطح کاربر به منابع حساس (مثل حافظه ، I/O) جلوگیری می‌کند. این جداسازی از وقوع حملات مخرب و اشتباهات تصادفی جلوگیری می‌کند.

سوال ۲.۱

به سوالات زیر پاسخ کوتاه بدهید.
الف)مراحل و مکانیزم پایه‌ای وقفه‌ها را بیان کنید و بگویید در سیستم‌عامل‌های مدرن، روند پاسخ دهی به وقفه چه تغییری کرده است؟

1. مکانیزم پایه‌ای وقفه‌ها (Interrupts):

وقفه‌ها مکانیزمی هستند که به CPU اجازه می‌دهند تا به رویدادهای ناگهانی و غیرهمزمان پاسخ دهد. این وقفه‌ها به وسیله سخت‌افزار یا نرم‌افزار تولید می‌شوند و وقتی رخ می‌دهند، اجرای فعلی CPU متوقف می‌شود و CPU به جای اجرای دستورالعمل‌های جاری، به اجرای کد مربوط به وقفه (Interrupt Handler) می‌پردازد.

مراحل اصلی مکانیزم وقفه‌ها به شرح زیر است:

- **1. تولید وقفه:** وقتی یک دستگاه آماده سرویس‌دهی است یا یک رویداد ناگهانی (مانند ورود داده از کیبورد یا تکمیل یک عملیات I/O) رخ می‌دهد، یک وقفه تولید می‌شود.
- **2. ذخیره وضعیت CPU: CPU** اجرای فعلی را متوقف می‌کند و وضعیت (حافظه و رجیسترهای فعلی) را ذخیره می‌کند تا بعداً بتواند به آن برگردد.
- **3. فراخوانی Interrupt Handler: CPU** به Interrupt Vector Table مراجعه می‌کند تا آدرس مربوط به کد وقفه را پیدا کند و آن را اجرا می‌کند.
- **4. بازگشت به پردازش:** بعد از اتمام کد وقفه، CPU وضعیت قبلی را بازیابی کرده و اجرای برنامه اصلی را از همان نقطه ادامه می‌دهد.

2. تغییرات در سیستم‌عامل‌های مدرن:

1. توانایی به تعویق انداختن پردازش وقفه‌ها در پردازش‌های بحرانی:

در سیستم‌های مدرن، وقتی CPU در حال اجرای بخش‌های بحرانی از کد است (مانند دسترسی به منابع مشترک)، ممکن است لازم باشد که پردازش وقفه به تعویق بیفتد تا اطمینان حاصل شود که منابع سیستم به طور منظم مدیریت می‌شوند. این فرآیند به معنای عدم پذیرش وقفه در زمان پردازش بحرانی است، اما وقفه‌ها در صف قرار می‌گیرند و پس از تکمیل این بخش بحرانی، پردازش می‌شوند.

2. راهکارهای بهینه برای اختصاص به Handler مربوط به هر دستگاه:

در سیستم‌عامل‌های قدیمی، ممکن بود وقفه‌ها به صورت دستی مدیریت شوند و کد وقفه برای هر دستگاه به‌طور مستقیم نوشته شود. اما در سیستم‌های مدرن، یک جدول وقفه‌ها (Interrupt Vector Table) وجود دارد که در آن برای هر دستگاه یک Handler تعریف شده است. CPU به طور خودکار پس از دریافت وقفه، به این جدول مراجعه می‌کند و Handler مربوطه را اجرا می‌کند، بدون اینکه نیازی به مداخله دستی باشد.

3. استفاده از وقفه‌های چند سطحی:

در سیستم‌های مدرن، وقفه‌ها اولویت‌بندی می‌شوند. به عنوان مثال، وقفه‌های بحرانی مانند خطاهای سخت‌افزاری (high-priority interrupts) باید سریع‌تر از وقفه‌های کم‌اهمیت (مانند تکمیل یک عملیات I/O) پردازش شوند. این وقفه‌های چند سطحی به سیستم اجازه می‌دهند که به وقفه‌های مهم‌تر سریع‌تر پاسخ دهد و وقفه‌های کم‌اهمیت را به تعویق بیندازد.

ب) DMA در پاسخ به چه نیازی استفاده می‌شود؟

DMA در پاسخ به نیاز به جابجایی حجم بالای داده‌ها با کمترین سربار پردازشی استفاده می‌شود. در سیستم‌های عمومی که شامل چندین دستگاه مختلف هستند، مدیریت I/O به دلیل اهمیت آن برای عملکرد و پایداری سیستم و همچنین تنوع دستگاه‌ها بخش بزرگی از کد سیستم‌عامل را تشکیل می‌دهد.

نیاز به DMA:

در روش‌های سنتی **interrupt-driven I/O**، هر زمان که داده‌ای برای ارسال یا دریافت وجود داشت، وقفه‌ای به CPU ارسال می‌شد و CPU باید مستقیماً با دستگاه I/O در ارتباط باشد تا داده‌ها را به حافظه منتقل کند. این روش برای جابجایی حجم کم داده‌ها مناسب است، اما در انتقال حجم زیادی از داده‌ها (مثل عملیات I/O برای دستگاه‌های ذخیره‌سازی حجیم مانند دیسک‌های NVMe سربار بسیار زیادی روی CPU ایجاد می‌کند و باعث کاهش کارایی سیستم می‌شود. اینجاست که **DMA** وارد عمل می‌شود.وقتی حجم زیادی داده باید انتقال پیدا کند، داده به صورت مستقیم از دیوایس کنترلر به حافظه اصلی یا از حافظه ی اصلی به دیوایس کنترلر بدون دخالت CPU انتقال پیدا میکند.

ج)چگونه پردازنده با دستگاه ارتباط برقرار کرده و یک عملیات DMA را اجرا میکند؟

1. شروع ارتباط CPU با دستگاه (راه‌اندازی عملیات):

- **درخواست انتقال داده:** پردازنده برای آغاز یک عملیات DMA، ابتدا یک درخواست I/O از طرف دستگاه دریافت می‌کند. مثلاً دستگاه ذخیره‌سازی (مانند دیسک) یا کارت شبکه نیاز به ارسال یا دریافت داده دارد.
- **پیکربندی کنترل‌کننده DMA:** پردازنده پس از دریافت درخواست انتقال داده، اطلاعات مورد نیاز برای عملیات DMA را به **DMA controller** ارسال می‌کند. این اطلاعات شامل موارد زیر است:
  - آدرس شروع حافظه‌ای که قرار است داده‌ها از آن خوانده یا در آن نوشته شود.

- مقدار داده‌هایی که باید منتقل شوند (تعداد بایت‌ها).
- جهت انتقال (از دستگاه به حافظه یا از حافظه به دستگاه).
- آدرس دستگاه ورودی/خروجی مورد نظر برای انتقال داده.

2. واگذاری عملیات به DMA controller:

- **کنترل‌کننده DMA عملیات را به عهده می‌گیرد:** پس از پیکربندی، CPU کنترل عملیات را به کنترل‌کننده DMA واگذار می‌کند و به اجرای سایر وظایف می‌پردازد. DMA controller به طور مستقیم به حافظه و دستگاه I/O دسترسی دارد و شروع به انتقال داده‌ها بین دستگاه و حافظه می‌کند.
- **انتقال داده‌ها به صورت مستقیم:** DMA controller داده‌ها را بدون نیاز به مداخله CPU مستقیماً از دستگاه به حافظه یا از حافظه به دستگاه انتقال می‌دهد. این فرآیند به طور کاملاً خودکار انجام می‌شود.

د) چگونه پردازنده از اتمام عملیات با خبر می‌شود؟

وقفه پس از تکمیل عملیات: هنگامی که انتقال داده‌ها به پایان رسید، DMA controller یک **وقفه (interrupt)** به پردازنده ارسال می‌کند تا آن را از اتمام عملیات مطلع سازد. این وقفه به CPU اطلاع می‌دهد که انتقال با موفقیت انجام شده و اکنون می‌تواند به پردازش داده‌های منتقل‌شده بپردازد.

سوال ۳.۱

(با توجه به پاسخ سوالات بالا، به این دو مورد پاسخ کامل دهید.)

الف)برای اسکن کردن یک عکس با حجم کم در حالت عادی چه مراحلی بین CPU ، memory و I/O طی میشود؟
در صورتی که حجم فایل یا عکس اسکن شده زیاد باشد چه روشی را پیشنهاد میکنید؟ این روش را به صورت کامل شرح دهید و نشان دهید چگونه تعداد دستورالعمل های اجرایی را کاهش میدهد.

الف) مراحل اسکن کردن یک عکس با حجم کم در حالت عادی بین CPU، memory و I/O:

در این حالت، عملیات I/O به صورت معمولی و بدون استفاده از DMA انجام می‌شود. این مراحل به شرح زیر است:

1. **دریافت درخواست اسکن از کاربر:**
   - کاربر از طریق برنامه کاربردی (application program) درخواست اسکن را ارسال می‌کند.

2. **ارسال دستور اسکن به دستگاه I/O (اسکنر):**
   - CPU با ارسال یک دستور به **دستگاه I/O** (در اینجا اسکنر)، عملیات اسکن را آغاز می‌کند.
3. **انتظار برای آماده شدن اسکنر:**
   - CPU منتظر می‌ماند تا اسکنر داده‌های اسکن‌شده را آماده کند.
4. **انتقال داده از اسکنر به حافظه اصلی (Memory):**
   - هنگامی که اسکنر داده‌های اسکن‌شده را آماده کرد، یک **وقفه (interrupt)** به CPU ارسال می‌شود تا CPU از آماده بودن داده‌ها مطلع شود.
   - CPU سپس داده‌های اسکن‌شده را مستقیماً از اسکنر به **حافظه اصلی (Memory)** انتقال می‌دهد. این انتقال از طریق دستورالعمل‌های I/O که توسط CPU اجرا می‌شود، صورت می‌گیرد.
5. **پردازش داده‌های اسکن‌شده:**
   - CPU پس از دریافت داده‌ها از اسکنر، آنها را پردازش می‌کند و نتایج را به کاربر نمایش می‌دهد یا در حافظه ذخیره می‌کند.

ب) حجم بالای فایل یا عکس اسکن‌شده:

در صورتی که فایل اسکن‌شده حجم بالایی داشته باشد، روش معمولی انتقال داده‌ها که شامل پردازش مداوم توسط CPU است، می‌تواند سربار زیادی برای سیستم ایجاد کند و کارایی را به شدت کاهش دهد. **روش پیشنهادی برای حل این مشکل استفاده از DMA (Direct Memory Access)** است. این روش به CPU اجازه می‌دهد که بدون دخالت مستقیم در انتقال داده‌های حجیم، وظایف خود را ادامه دهد.

شرح کامل روش DMA:

1. **دریافت درخواست اسکن:**
   - همانند حالت عادی، کاربر درخواست اسکن یک عکس حجیم را ارسال می‌کند.
2. **پیکربندی کنترل‌کننده DMA:**
   - به جای اینکه CPU مستقیماً داده‌های اسکن‌شده را از اسکنر دریافت کند، **DMA controller** را برای انجام این کار پیکربندی می‌کند. CPU اطلاعات لازم مانند **آدرس حافظه**، **اندازه داده‌های قابل انتقال** و **جهت انتقال داده‌ها** را به DMA controller ارسال می‌کند.
3. **انتقال داده‌ها توسط DMA:**
   - پس از پیکربندی، **DMA controller** مستقیماً داده‌ها را از اسکنر به حافظه اصلی منتقل می‌کند، بدون اینکه CPU در این فرآیند دخالت داشته باشد. این کار به صورت خودکار انجام می‌شود.
4. **وقفه (Interrupt) در پایان انتقال:**
   - پس از پایان انتقال داده‌ها، **DMA controller** با ارسال یک **وقفه** به CPU اطلاع می‌دهد که داده‌ها با موفقیت به حافظه اصلی منتقل شده‌اند.
5. **پردازش داده‌ها توسط CPU:**

- CPU پس از دریافت وقفه، داده‌های اسکن‌شده را پردازش کرده و آنها را به کاربر نمایش می‌دهد یا ذخیره می‌کند.

چگونه DMA تعداد دستورالعمل‌های اجرایی را کاهش می‌دهد:

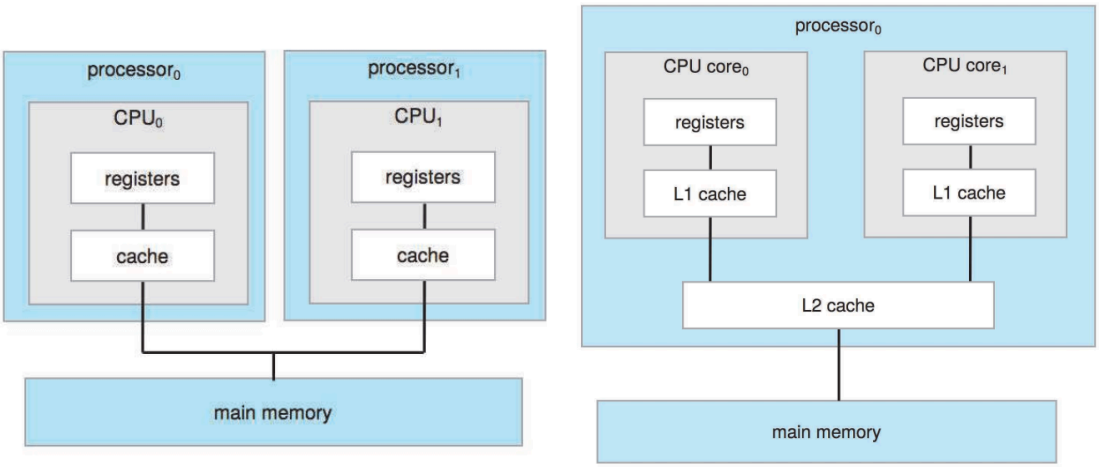
- **بدون DMA:** در روش معمولی، CPU باید به طور مداوم داده‌ها را از اسکنر دریافت کرده و آنها را به حافظه انتقال دهد. هر بایت یا بلاک داده نیاز به پردازش مستقیم توسط CPU دارد که تعداد زیادی دستورالعمل را مصرف می‌کند.
- **با DMA:** با استفاده از CPU ،DMA تنها در ابتدا برای پیکربندی DMA controller و در انتها برای دریافت وقفه دخالت دارد. تمام مراحل انتقال داده توسط DMA controller انجام می‌شود و این امر به کاهش قابل توجه تعداد **دستورالعمل‌های CPU** منجر می‌شود.

ب)آیا عملیات همزمان اجرای برنامه توسط CPU و انتقال داده توسط DMA ممکن است تداخلی با هم داشته باشند؟در صورت مثبت بودن جواب این اتفاق در چه شرایطی رخ میدهد؟ بین DMA و CPU اولویت اصلی دسترسی به حافظه با کدام است؟چرا؟

ممکن است که دستورات اجرا شونده توسط پردازنده نیاز به دسترسی به حافظه اصلی داشته باشند. در اینصورت DMA و پردازنده نمی توانند به طور همزمان از باس متصل به حافظه اصلی استفاده کنند و با هم تداخل پیدا میکنند. در این حالت معمولا اولویت دسترسی به حافظه با DMA است تا بتواند انتقال بین بافرهای O/I و حافظه ی اصلی را به انجام برساند و اطلاعات از دست نرود.

سوال ۴.۱

نحوه‌ی اشتراک یا جدا بودن منابع مختلف را در ساختار سیستم‌های چند پردازنده و چندهسته‌ای مقایسه کرده و نکات مثبت و منفی هر سیستم را بیان کنید.



1. سیستم‌های چند پردازنده‌ای (Multiprocessor Systems):

این سیستم‌ها شامل چندین پردازنده فیزیکی هستند که هر پردازنده یک یا چند هسته دارد. به طور معمول، پردازنده‌ها منابع زیر را به اشتراک می‌گذارند:

- **حافظه فیزیکی (RAM):** تمام پردازنده‌ها به یک حافظه فیزیکی مشترک دسترسی دارند.
- **سیستم باس:** تمام پردازنده‌ها از طریق یک باس سیستم به حافظه و سایر دستگاه‌ها متصل هستند.
- **دستگاه‌های جانبی:** دستگاه‌های I/O مانند دیسک‌های سخت و کارت‌های شبکه به‌صورت مشترک استفاده می‌شوند.

نکات مثبت:

- **افزایش توان محاسباتی:** به ازای هر پردازنده جدید، توان پردازشی سیستم افزایش می‌یابد و می‌توان وظایف بیشتری را به‌طور همزمان اجرا کرد.
- **اجرای چندین فرایند همزمان:** امکان اجرای همزمان چندین فرایند بدون افت عملکرد زیاد.

نکات منفی:

- **مشکل رقابت برای منابع مشترک:** هنگامی که تعداد پردازنده‌ها افزایش می‌یابد، دسترسی همزمان به حافظه و سیستم باس باعث کاهش کارایی و افزایش **"رقابت برای منابع"** می‌شود. این مسئله به‌ویژه در سیستم‌های با تعداد زیاد پردازنده‌ها مشکل‌ساز است.
- **سربار ارتباطی:** برای هماهنگی بین پردازنده‌ها و دسترسی به حافظه مشترک، سربار ارتباطی ایجاد می‌شود که باعث کاهش کارایی می‌گردد.

---

2. سیستم‌های چندهسته‌ای (Multicore Systems):

در این سیستم‌ها، چندین هسته پردازشی بر روی یک تراشه فیزیکی (پردازنده) قرار دارند. هسته‌ها برخی منابع را به اشتراک می‌گذارند:

- **حافظه کش:** هر هسته دارای یک **کش سطح 1 (L1)** اختصاصی است که سریع و کوچک است. **کش سطح 2 (L2)** ممکن است به صورت مشترک بین هسته‌ها قرار گیرد.
- **حافظه فیزیکی:** تمام هسته‌ها به حافظه فیزیکی مشترک دسترسی دارند که به وسیله یک باس داخلی (روی تراشه) انجام می‌شود.

- **سیستم باس:** باس داخلی (درون تراشه‌ای) سریع‌تر و کاراتر از باس بین پردازنده‌ای در سیستم‌های چند پردازنده‌ای است.

نکات مثبت:

- **ارتباط داخلی سریع‌تر:** به دلیل استفاده از باس داخلی (روی تراشه)، ارتباط بین هسته‌ها و حافظه سریع‌تر از ارتباط بین پردازنده‌ها در سیستم‌های چند پردازنده‌ای است.
- **کاهش مصرف انرژی:** تراشه‌های چندهسته‌ای به دلیل اینکه چندین هسته روی یک تراشه قرار دارند، **کمتر انرژی مصرف می‌کنند** نسبت به داشتن چندین پردازنده مجزا.
- **افزایش کارایی در عملیات همزمان:** به دلیل به اشتراک‌گذاری منابع و ارتباط سریع‌تر بین هسته‌ها، اجرای فرآیندهای همزمان کاراتر است.

نکات منفی:

- **محدودیت فیزیکی:** به دلیل محدودیت فیزیکی تراشه، تعداد هسته‌ها در یک پردازنده محدود است و این باعث محدودیت در توان پردازشی سیستم می‌شود.
- **تاثیر استفاده مشترک از حافظه کش:** اگر کش سطح 2 (L2) یا سطوح بالاتر به صورت مشترک استفاده شود، رقابت بین هسته‌ها برای دسترسی به حافظه کش ممکن است باعث کاهش کارایی شود.

مقایسه نهایی:

ویژگی	سیستم‌های چند پردازنده‌ای	سیستم‌های چندهسته‌ای
حافظه	حافظه فیزیکی و منابع مشترک برای تمام پردازنده‌ها	حافظه فیزیکی مشترک، کش سطح 1 جدا و کش سطح 2 مشترک
ارتباط داخلی	کندتر به دلیل استفاده از سیستم باس	سریع‌تر به دلیل باس داخلی روی تراشه
مصرف انرژی	بیشتر به دلیل وجود پردازنده‌های مستقل	کمتر به دلیل قرار گرفتن چند هسته روی یک تراشه
مقیاس‌پذیری	با افزایش تعداد پردازنده‌ها کارایی کاهش می‌یابد	مقیاس‌پذیر اما محدود به تعداد هسته‌ها روی یک تراشه

کاربردهای معمول	سرورها و سیستم‌های قدرتمند برای پردازش‌های سنگین موبایل‌ها، لپ‌تاپ‌ها، سرورهای مدرن

جمع‌بندی:

- **سیستم‌های چند پردازنده‌ای** برای کاربردهایی که نیاز به پردازش موازی بالا دارند مناسب هستند، اما با افزایش تعداد پردازنده‌ها به دلیل رقابت برای منابع مشترک مانند حافظه و باس، کارایی کاهش می‌یابد.
- **سیستم‌های چندهسته‌ای** با توجه به ارتباط داخلی سریع‌تر و مصرف انرژی کمتر، مناسب‌تر برای دستگاه‌های موبایل و لپ‌تاپ‌ها هستند، اما محدودیت فیزیکی تراشه باعث محدود شدن تعداد هسته‌ها می‌شود.

سوال۱.۲

الف)درباره دسته بندی system call ها تحقیق کنید و از هر دسته بندی یک system call نام برده و کاربرد آن را بصورت کوتاه بنویسید.

system call ها بطور کلی به 6 دسته تقسیم می شوند.

- **process control** - نمونه : fork() - ساخت یک پروسس جدید
- **file management** - نمونه: read() - خواندن فایل
- **device management** - نمونه: ioctl() - عملیات ورودی/خروجی مخصوص Device
- **information maintenance** - نمونه: getpid() - گرفتن pid مربوط به پروسس کنونی
- **Communications** - نمونه: pipe () - ساخت یک pipe (ارتباط بین دو process)
- **Protection** - نمونه: chmod() - تعیین سطوح دسترسی به فایل

ب) درباره system call های زیر تحقیق کنید و یک توضیح کوتاه درباره کارکرد و کاربرد آنها(حداکثر دو خط) بنویسید.

- **Fork** : ساخت یک پروسس جدید که در حالت عادی ادامه برنامه والد خود را اجرا می کند.
- **Exit** : پایان دادن به اجرای یک process، آزاد کردن منابع تخصیص یافته به آن و بازگردانی آنها سیستم عامل.
- **Chmod**: تعیین سطوح دسترسی به فایل های سیستم. (read, write, execute)

ج) بطور کوتاه بنویسید هریک از توابع نام برده شده(از مجموعه کتابخانه های استاندارد زبان C) چه کاری انجام می دهد و برای هر تابع system call های مهمی که طی آن فراخوانی می شود را نام ببرید.

- scanf()

دریافت ورودی از کاربر.

read

- **printf()**

نمایش خروجی در کنسول

write

- **malloc()**

تخصیص حافظه دینامیک در **heap**

mmap

- **fopen()**

باز کردن یک فایل. این تابع یک **file descriptor** برمی‌گرداند.

open

سوال ۲.۲

الف) هر یک از ساختارهای میکرو کرنل، لایه ای ، یکپارچه(**monolithic**) و **modular** را از نظر نحوه پیاده سازی، میزان کارایی، و میزان انعطاف پذیری بررسی کنید.

- سیستم عامل **monolithic** هیچ ساختاری ندارد. همه عملکردهای کرنل در یک فایل باینری استاتیک واحد ارائه می شود. اگرچه اصلاح چنین سیستم هایی دشوار است و انعطاف پذیری پایینی دارند، اما مزیت اصلی آنها کارایی بالا است.
- سیستم عامل **layered** به تعدادی لایه گسسته تقسیم می شود که لایه پایینی رابط سخت افزاری و بالاترین لایه رابط کاربری است. در این ساختار، در هر لایه تعداد کمی از **function** های کرنل پیاده سازی شده و لایه ای بودن این سیستم ها باعث انعطاف پذیری نسبتا بالای آنها می شود. اگرچه سیستم‌های لایه‌ای تا حدی موفقیت‌آمیز بوده‌اند، این رویکرد به دلیل کارایی پایین و سربار بیش از حد در ارتباط لایه ها، عموماً برای طراحی سیستم‌عامل ایده‌آل نیست.

- سیستم عامل **microkernel**، از انعطاف پذیری بالایی برخوردار است . کرنل این سیستم ها بطور **minimal** طراحی شده و مابقی کارایی های سیستم در سطح کاربر پیاده سازی شده اند. این ساختار برای تبادل اطلاعات و ایجاد ارتباط از **messaging** استفاده می کند. ارسال این پیام های تبادلی و **switch** کردن بین **process** های مختلف سبب ایجاد **overhead** بالایی می شود و در نتیجه از کارایی آن کاسته می شود .

- در سیستم عامل های **modular** ، خدمات سیستم عامل از طریق ماژول هایی ارائه می شود که می توانند در طول زمان اجرا بارگذاری و حذف شوند. بسیاری از سیستم عامل های معاصر به صورت ترکیبی از **monolithic** و **modular** ساخته می شوند. به این ترتیب از کارایی سیستم بالای سیستم یکپارچه استفاده کرده و با استفاده از رویکرد **modular** انعطاف پذیری آن تامین می شود.

ب) در ساختار میکرو کرنل ارتباط **user program**(برنامه سطح کاربر) با **system call** ها از چه طریقی صورت می گیرد؟

Message passing

سوال ۳.۲

الف) تفاوت های **emulator** و **virtual machine** را بیان کنید.

Emulation شامل شبیه سازی سخت افزار کامپیوتر در نرم افزار است و معمولا زمانی استفاده می شود که نوع **CPU** فعلی با نوع **CPU** هدف متفاوت باشد. در حالی که **virtualization** برای اجرای سیستم عاملی منطبق با یک **CPU** بر روی سیستم عاملی دیگر بر روی همان **CPU** است. در واقع **virtualization** محیط نرم افزاری و در **emulation** یک محیط سخت افزاری شبیه سازی می شود.

ب) اگر یک برنامه ویندوزی را بر روی لینوکس اجرا کنیم از **emulation** بهره برده ایم یا **virtualization**؟ چرا؟

از **virtualization** بهره برده ایم. زیرا صرفا برای اینکه برنامه ویندوزی بتواند اجرا شود، روی محیط لینوکس محیط نرم افزاری ویندوز را شبیه سازی کرده ایم و شبیه سازی سخت افزار در این فرایند انجام نمی شود.

ج) **Docker** چیست؟ درباره مزایای آن نسبت به **VM** تحقیق کنید .

Docker یک پلتفرم اپن سورس است که به توسعه‌دهندگان اجازه می‌دهد برنامه‌ها را در قالب **کانتینر** اجرا، توسعه و مدیریت کنند. کانتینرها بسته‌های سبک و قابل حمل هستند که شامل کد برنامه و تمام وابستگی‌های مورد نیاز برای اجرای آن برنامه می‌باشند. این قابلیت به توسعه‌دهندگان اجازه می‌دهد برنامه‌های خود را به راحتی در محیط‌های مختلف بدون نگرانی از ناسازگاری اجرا کنند.

مزایای Docker نسبت به VM :

مصرف منابع کمتر:

کانتینرها فقط شامل برنامه و وابستگی‌های آن هستند و از هسته (**kernel**) سیستم‌عامل میزبان استفاده می‌کنند. در مقابل، ماشین‌های مجازی هر کدام یک سیستم‌عامل کامل دارند، که منجر به مصرف بیشتر منابع (**CPU**، **RAM** و دیسک) می‌شود.

سرعت بالاتر:

استارت‌آپ کانتینرها به دلیل حجم کمتر و عدم نیاز به بوت سیستم‌عامل جدید، بسیار سریع‌تر از ماشین‌های مجازی است. این باعث می‌شود اجرای و مقیاس‌پذیری برنامه‌ها سریع‌تر و آسان‌تر باشد.

قابلیت حمل بالا (Portability):

کانتینرهای Docker بدون توجه به محیط (سیستم‌عامل میزبان) به راحتی می‌توانند اجرا شوند. این یعنی شما می‌توانید یک کانتینر را از محیط توسعه به محیط تولید یا بین سرورها جابجا کنید، بدون اینکه نگرانی از ناسازگاری‌ها داشته باشید.

سوال ۴.۳

الف) LKM چیست و به طور معمول چه کاربردی در سیستم دارد ؟

LKM = Loadable Kernel Module

سیستم عامل لینوکس از LKM ها، در درجه اول برای پشتیبانی از Device Driver ها و file system ها استفاده می کند.

ب) تفاوت اصلی در اضافه کردن یک system call و LKM به سیستم عامل چیست؟

برای افزودن یک سیستم‌کال جدید، kernel سیستم عامل باید مجدداً کامپایل و اجرا شود و سیستم مجددا راه اندازی شود. درحالی که LKM به طور پویا به kernel ، بدون اینکه نیاز به کامپایل kernel یا راه‌اندازی مجدد سیستم داشته باشیم، اضافه یا از آن حذف می‌شود، .

ج) برای دسترسی به لیست module های سیستم دستوری نام ببرید.

lsmod

د) درباره تابع ioctl تحقیق کنید و توضیح دهید در یک LKM چه امکانی را فراهم می کند؟

input/output control

همانطور که می دانیم در سیستم شامل دو فضای kernel space و user space است. فضای کرنل به شدت حفاظت شده است. در مقابل، فضای کاربر، ناحیه‌ای از حافظه است که همه برنامه‌های حالت کاربر در آن کار می‌کنند و کاربر به راحتی میتواند با آن تعامل داشته باشد.

Ioctl یک system call برای ایجاد ارتباط بین فضای کرنل و فضای کاربر است که در فضای کاربر با تابع ioctl() در دسترس است . به این ترتیب، این تابع در lkm ها برای ارتباط از سطح کاربر با device driver ها در سطح کرنل استفاده می شود.

(https://embetronicx.com/tutorials/linux/device-drivers/ioctl-tutorial-in-linux/)

ه) دستورهای insmod , rmmod چه کاری انجام می دهند؟

insmod: برای بارگذاری (install کردن) یک **LKM** به **kernel** استفاده می‌شود. به این معنی که ماژول را به **kernel** سیستم اضافه می‌کند.

rmmod: برای حذف یک ماژول بارگذاری شده از **kernel** استفاده می‌شود.

و) تفاوت توابع ()printf و ()printk چیست؟

printf: این تابع در سطح کاربران (**user space**) استفاده می‌شود و خروجی را به استاندارد خروجی (مانند کنسول یا فایل) می‌فرستد.

printk: این تابع در سطح **kernel** برای نوشتن پیام‌های دیباگ یا ثبت **log** استفاده می‌شود.

تمام پیام‌های **printk**() در یک بافر در **kernel** چاپ می‌شوند، که از طریق **/dev/kmsg** به **userspace** صادر می‌شود. برای خواندن آن می توان از دستور **dmesg** در کنسول استفاده کرد.