

فصل

۲

ساختمان‌های سیستم کامپیووتر

قبل از بررسی تفصیلی عملکرد سیستم، نیازمند دانش کلی از ساختمان سیستم کامپیووتری، هستیم. در این فصل، منظور یادآوری، به بررسی پیش زمینه اجزای ساختمان و معماری کامپیووتر می‌پردازیم. و کسانی که این مشاهیم را می‌دانند، می‌توانند فصل را رد کنند. چون سیستم عامل ارتباط نزدیک با مکانیزم I/O کامپیووتر دارد، ابتدا این موضوع بحث می‌شود. بخش‌های بعدی ساختمان ذخیره داده را دربرمی‌گیرند.

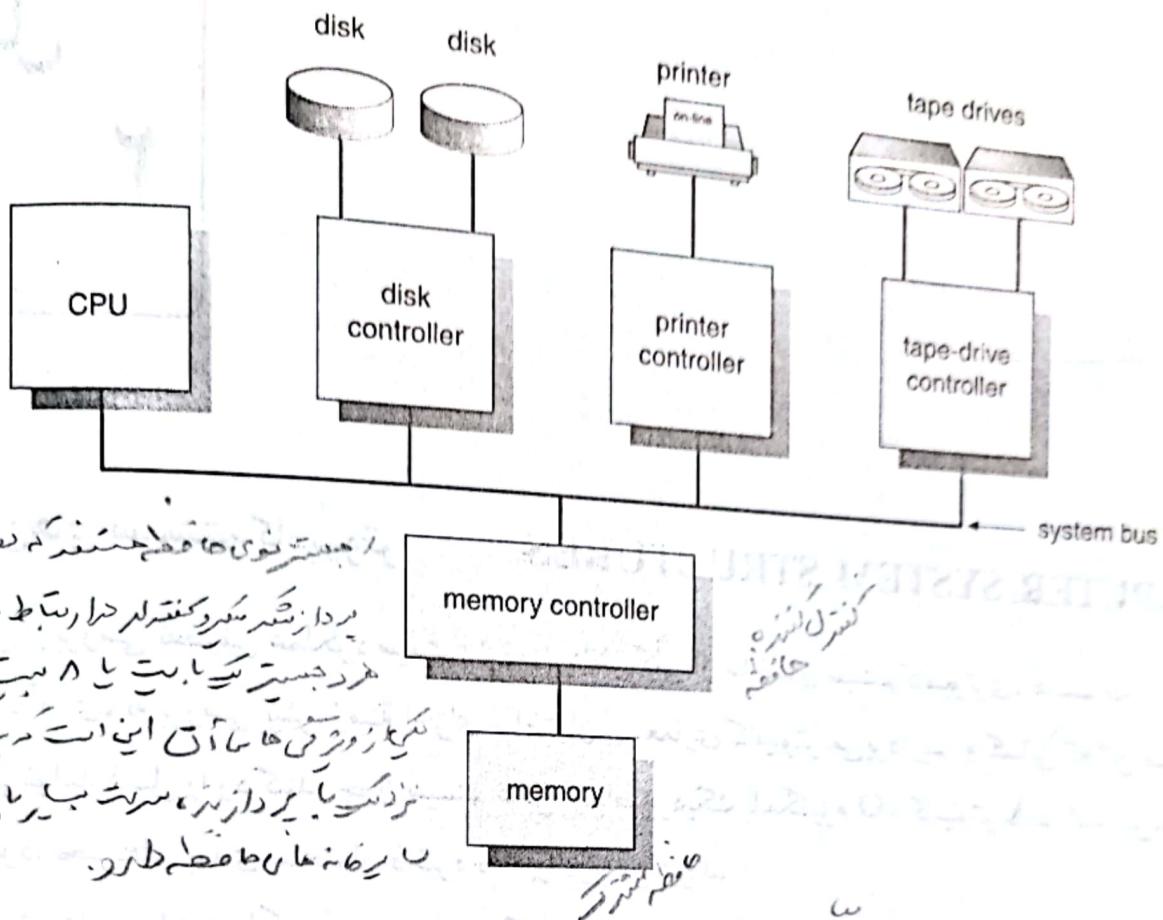
سیستم عامل، باید عملکرد صحیح سیستم کامپیووتری را تضمین نماید. طوری که برنامه‌های کاربر نباید با کار درست سیستم تعارض داشته باشند و سخت‌افزار باید مکانیزم مناسبی جهت تضمین رفتار درست، فراهم آورد. در انتها این فصل، معماری کامپیووتر پایه را شرح خواهیم داد که در کسیستم عامل تابعی را ممکن خواهد ساخت.

2.1. Computer System Operation

۱. عمل سیستم کامپیووتر

یک سیستم کامپیووتری همه - منظوره مدرن، شامل یک CPU و تعدادی کنترل‌کننده‌های دستگاه است که از طریق یک گذرگاه ویژه که دستیابی به حافظه مشترک را ممکن می‌سازد، بهم مربوط می‌شوند (شکل ۱-۲). هر کنترل‌کننده دستگاه، مربوط به یک نوع خاص از وسایل می‌باشد (به عنوان مثال، گردنده‌های دیسک، دستگاه‌های صوتی و صفحه نمایش ویدیویی). CPU و کنترل‌کننده‌های دستگاه می‌توانند هم‌روند اجرا شوند، و برای سیکل‌های اجرف حافظه رقابت داشته باشند. جهت تضمین دسترسی منظم به حافظه مشترک، یک کنترل‌کننده حافظه فراهم شده است که

1. memory cycles



شکل ۱-۲: یک سیستم کامپیوتری مدرن

وظیفه آن، سنکرون کردن دستیابی به حافظه است.

یک سیستم در شروع اجرا - وقتی که روشن می‌شود و یا ری‌بوت^۱ می‌شود - به برنامه اولیه‌ای جهت اجرا نیازمند است. این برنامه اولیه، برنامه ساده آغازگر^۲ می‌باشد. این برنامه به تمام اجزاء سیستم، از رجیسترهای CPU تا کنترل کننده‌های دستگاه مقادیر آغازین می‌دهد. برنامه آغازگر، می‌داند که سیستم عامل را چگونه بار نماید و چگونه به اجراء درآورد. بدین منظور، هسته^۳ سیستم عامل را یافته و در حافظه بار می‌نماید. سیستم عامل، آن گاه، شروع به اجرای اولین پردازش^۴، به نام «init»^{*} نموده و منتظر وقوع حوادثی می‌ماند. وقوع یک حادثه، معمولاً سیگنال وقفه از سخت‌افزار یا وقفه نرم‌افزاری است. (سخت‌افزار، یک اینترپریت را به طور آسنکرون در هر زمان، توسط ارسال سیگنالی به CPU تریگر^۵ می‌کند و نرم‌افزار، توسط اجرای دستور العمل خاصی به نام فراخوانی سیستم (یا به نام

- 1. reboot راه‌اندازی مجدد
- 2. bootstrap programs
- 3. kernel
- 4. process

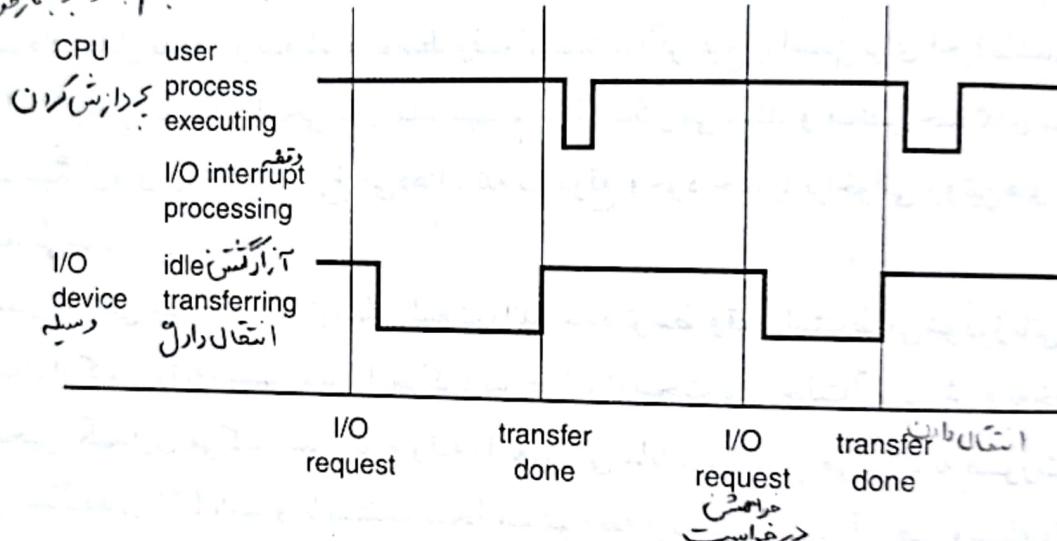
* این مطلب در مورد بستم بونیکس صادق است.

راه انداختن

فرآخوانی مانیتور) می‌تواند موجب تریگر کردن و قله گردد.

نوع گوناگون حوادث جهت تریگر کردن و قله وجود دارند - به عنوان مثال تکمیل عمل I/O، عمل تقسیم بصر، دسترسی حافظه نامعتبر، و درخواست سرویس سیستم عامل. برای هر چنین وقنه‌ای، روتین سرویس دهنده‌ای زمام شده است. زمانی که CPU وقنه دریافت می‌کند، آنچه را که در حال انجام بود، متوقف نموده و به مکان ثابتی، جهت اجرا، انتقال می‌یابد. مکان ثابت، معمولاً شامل آدرس شروع روتین‌های سرویس دهنده^۱ وقنه می‌باشد. روتین سرویس دهنده اجرا می‌شود؛ در خاتمه، CPU برنامه وقنه دیده را، از سر می‌گیرد که اگر زمانی این عمل در شکل

(۲-۲) نمایش داده شده است. وقنه شل می‌دلگیر؛ CPU است. دفعی کاربر CPU کار را بامی دهد تا اتفاق می‌افتد وقنه این کار را در حال پردازش متصوّف نموده باشد آدرس آفرینی سطر در جای ذخیره کند بعد از این مملیات وقنه آربیت تصدیع کند. به عبارت ترین کار در حال اجرامتصوّف دستگاه خارجی را از این وقنه به تدارک آزمیم، و بعد بارخورس می‌رسد.



شکل ۲-۲: دیاگرام زمانی وقنه برای پردازش ساده خروجی

وقنه‌ها، جزء مهمی از معماری کامپیوتر هستند. هر سیستم کامپیوتری، مکانیزم وقنه خود را دارد، ولی توابع شلودی مشترکند. وقنه با استی کنترل را به روای سرویس دهنده وقنه مناسبی، انتقال دهد. وقنه با استی سریعاً پاسخ داده شود و با فرض این که تعداد پیش تعريف شده‌ای وقنه‌های ممکن، وجود دارند، جدولی از اشاره گرها به روتین‌های سرویس، هم می‌تواند استفاده شود. در این حالت، فرآخوانی سرویس به صورت غیرمستقیم رخ می‌دهد. معمولاً، جدول نزدیک در حافظه پایین نگهداری می‌شود. این آرایه، یا بردار وقنه^۲، بر حسب شماره یگانه دستگاه، که از درخواست وقنه دریافت می‌شود، اندیس گذاری می‌گردد. سیستم‌های عامل MS-DOS و یونیکس بدین طریق وقنه را دیس پیج^۳ می‌کنند.

5. trigger راه انداختن

3. dispatch

1. interrupt service routine

وقنه

2. interrupt vector

۳۰ فصل ۲: ساختمان‌های سیستم کامپیوتر

آدرس اندراپت را در شیوه ذفر را لش.

معماری و ققهه باستی آدرس دستورالعمل اینترپت شده را نگه دارد که در سیستم‌های قدیمی در مکان ثابتی و در سیستم‌های امروزی در پشتِ^۱ قرار می‌گیرد^۲. اگر وققه، حالت پردازش را عوض نماید، مانند رجیسترها، باستی آنها را قبلًا ذخیره و بعداً بازیابی نماید یا این عمل در سخت‌افزار صورت گیرد. بعد از اتمام سرویس، آدرس بازگشت در شمارنده برنامه (رجیستر PC) لود شده و محاسبات از سر گرفته می‌شود.

معمولًاً سایر وققه‌ها در زمان پردازش وققه، ممنوع می‌شوند و بعد از اتمام کار وققه اول، دوباره مجاز می‌گردند. اگر وققه تو در تو^۲ مجاز باشد، البته نیازی به این عمل نداریم. در معماری وققه پیشرفت، وققه‌ها دارای طرح اولویت هستند که بر حسب اهمیت نسبی آنها در نظر گرفته شده است. وققه با حق تقدم بالاتر، می‌تواند هر زمان رخ دهد ولی معمولًاً وققه‌های هم سطح یا پایین‌تر، پوشانده^۳ می‌شوند و یا انتخابی، ممنوع می‌شوند تا وققه‌های ناخواسته و غیرضروری رخ ندهد.

سیستم‌های عامل مدرن، رانده شده توسط وققه^۴ هستند. اگر هیچ پراسسی برای اجرا نباشد، هیچ وسیله I/O سرویس نخواهد، و هیچ کاربری پاسخی نخواهد، سیستم عامل یکار می‌شیند و منتظر حادثه‌ای می‌ماند. حوادث معمولًاً توسط سیگنال‌های وققه یا تله^۵ رخ می‌دهند. تله در موقع وجود خطای فراخوانی روتین‌های سیستم، توسط نرم‌افزار، تولید می‌شود.

ساختمان عمومی سیستم عامل، توسط، طبیعت رانده شده توسط وققه، استنباط می‌شود. زمانی که وققه (یا تله) رخ داد، سخت‌افزار کنترل را به سیستم عامل بازمی‌گرداند. در ابتدا، سخت‌افزار حالت^۶ پردازش و به خصوص شمارنده برنامه را در محلی نگهداری می‌کند. سپس نوع وققه را تعیین می‌نماید. این تعیین می‌تواند به صورت بازپرسی^۷ از تمامی کلاس دستگاه‌های I/O باشد و یا مستقیماً نتیجه سیستم وققه برداری^۸ که در آن هر وسیله دارای یک وققه منحصر به فرد است، باشد. برای هر نوع، قطعه کد جداگانه‌ای در سیستم عامل، نحوه کار را تعیین خواهد نمود.

۲.۲. ساختمان ورودی/خروجی

2.2. I/O Structure

همان گونه که در بخش ۲.۱ ذکر شد، یک سیستم کامپیوتری همه-منظوره، شامل یک CPU و تعدادی کنترل‌کننده دستگاه که از طریق یک گذرگاه مشترک متصل می‌شوند، می‌باشد. هر کنترل‌کننده دستگاه مربوط به یک نوع ویژه می‌باشد. بر حسب کنترل‌کننده، ممکن است بیش از یک دستگاه متصل به آن وجود داشته باشد. به عنوان مثال کنترل‌کننده واسطه سیستم کامپیوتری کوچک (SCSI)* می‌تواند ۷ یا بیشتر وسیله جانبی متصل به خود داشته باشد.

1. stack

2. nested

3. mask

4. interrupt driven

5. trap

6. state

7. polling

8. vectored interrupt system

* Small Computer system Interface

هر کنترل کننده‌ای تعدادی رجیستر خاص - منظوره و مقادیری بافر ذخیره محلی را دارد. کنترلر مسئول انتقال داده از وسیله به بافر خود می‌باشد اندازه بافر بر حسب وسیله کنترل کننده متفاوت است. به عنوان مثال، بافر کنترل کننده دیسک هم اندازه یا مضرب صحیحی از اندازه کوچک‌ترین بخش قابل آدرس دیسک، به نام قطاع^۱، و معمولاً ۵۱۲ بایت می‌باشد.

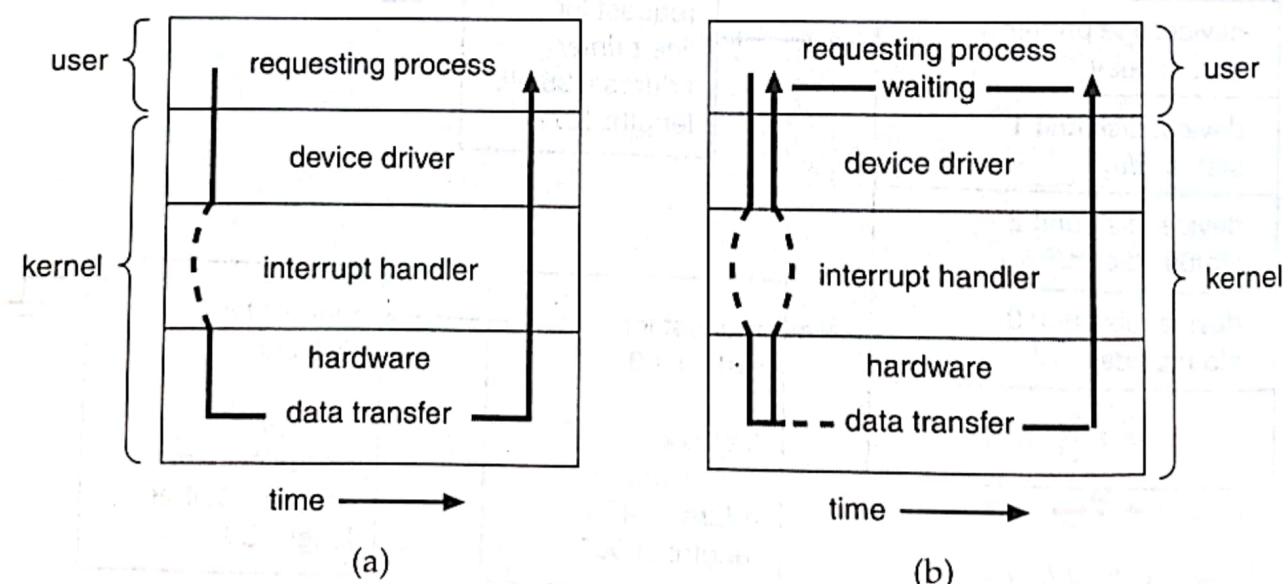
2.2.1. I/O Interrupt

I/O وقفه ۱.۲.۲

آغاز یک عمل I/O بدين گونه است که CPU رجیسترهاي مناسبی از کنترل کننده ابزار را بارگذاري می‌نماید. کنترل کننده، به نوعی خود، محتويات رجیسترها را تست نموده و عمل خاصی را انجام می‌دهد. مثلاً اگر درخواست خواندن باشد، شروع به انتقال داده از وسیله به بافر می‌نماید و در پایان CPU را از اتمام کار، آگاه می‌سازد. اين ارتباط از طریق تریگر کردن یک سیگنال وقفه، صورت می‌گیرد.

این وضعیت، معمولاً، به عنوان نتیجه درخواست I/O، توسط یک پراسس کاربر رخ خواهد داد. وقتی که I/O آغاز می‌شود، دو نحوه عمل ممکن است. در ساده‌ترین حالت، I/O آغاز می‌شود، در تکمیل I/O، کنترل به پراسس کاربر منتقل می‌شود این حالت به نام I/O سنکرون نامیده می‌شود. در نحوه دیگر، به نام I/O آسنکرون کنترل به برنامه کاربر بر می‌گردد بدون آن که منتظر تکمیل I/O باشد. سپس I/O ادامه می‌یابد در حالی که سایر اعمال سیستم در حال

اجرا هستند (شکل ۳-۲).



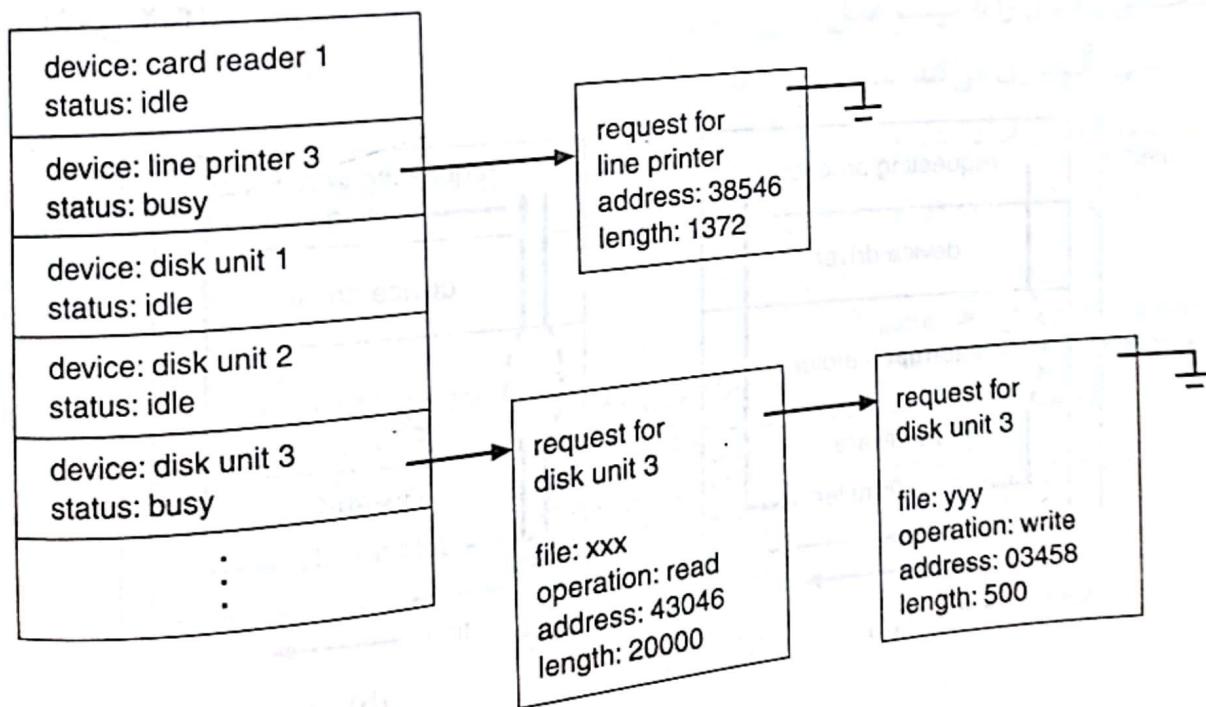
شکل ۳-۲: دو روش I/O: (a) سنکرون (آسنکرون) (b) حمزه از بافر

~~فرز~~ انتظار تکمیل I/O به یکی از دو روش صورت می‌گیرد. در بعضی سیستم‌ها، دستور خاصی به نام **wait** وجود دارد که CPU را بیکار می‌کند تا وقفه‌ای رخ دهد. ماشین‌هایی که این دستور العمل را ندارند می‌توانند یک لوب انتظار داشته باشند:

loop : jmp loop

این حلقه سفت ۱ تا رخداد وقفه، ادامه دارد. چنان‌لوپی در بازپرسی فلاغ دستگاه‌های I/O هم به کار می‌رود که از مکانیزم وقفه استفاده نمی‌کنند (I/O کنترل شده توسط برنامه).

اگر CPU همواره منتظر تکمیل I/O بماند، حداکثر یک درخواست I/O در هر زمان وجود خواهد داشت. لذا، هر بار که یک وقفه I/O رخ می‌دهد، سیستم عامل می‌داند که کدام وسیله در حال صدور وقفه است. این روش، عمل همروندی دستگاه‌های متعدد I/O را ناممکن ساخته و نیز امکان عمل روی هم افتدان محاسبه و I/O را منع می‌کند. آلترناتیو بهتر آن است که عمل I/O را آغاز نمود و پردازش را با یک پراسس سیستم عامل یا کاربر دیگر، ادامه داد. پس یک فراخوانی سیستم (درخواست سیستم عامل) در موقع انتظار I/O لازم است. سیستم عامل از جدولی به نام جدول حالت وسیله^۲ که یک ورودی^۳ به ازاء هر وسیله I/O دارد، استفاده می‌کند (شکل ۴-۲).



شکل ۴-۲: جدول وضعیت-وسیله

2. device-status table

3. entry

فرز را

هر ورودی شامل نوع وسیله، آدرس آن، و حالت آن (بیکار^۱، مشغول، عمل نمی‌کند) می‌باشد. اگر وسیله درخواستی داشته باشد، نوع درخواست و سایر پارامترها نیز ثبت می‌شوند. از آن جایی که پراسس‌های دیگری ممکن است درخواست دستگاه یکسانی را صادر نمایند، سیستم عامل، یک صفت انتظار^۲ - لیستی از درخواست‌های منتظر - برای هر وسیله تهیه می‌کند.

وسیله I/O با صدور وقفه، تقاضای سرویس می‌نماید. معمولاً وقفه، به معنای پایان عمل I/O است و سیستم عامل وضعیت دستگاه را عوض می‌نماید. حال پراسس منتظر از صفت انتظار برای وسیله، می‌تواند سرویس داده شود. بالاخره، کنترل از وقفه I/O خارج می‌شود. اگر پراسسی منتظر تکمیل این درخواست باشد (همان‌گونه که در جدول حالت وسیله ثبت شده است)، کحال می‌توان کنترل را به آن بازگرداند. در غیر این صورت، می‌توانیم به آنچه که قبل از وقفه I/O در حال انجام بودیم، برگردیم: به اجرای برنامه کاربر (برنامه یک عمل I/O را آغاز کرده و آن عمل اکنون تمام شده است، اما برنامه هنوز منتظر اتمام عمل نمانده است) یا به حلقة انتظار (برنامه دو یا چند عمل I/O آغاز کرده و منتظر یکی از آنها که خاص است می‌باشد ولی این وقفه، از سایر آنها آمده است). در یک سیستم اشتراک-زمانی، سیستم عامل می‌تواند به پراسس دیگری که منتظر اجرا^۳ است، سوئیچ نماید.

طرح‌های استفاده شده در بعضی وسائل ورودی، ممکن است با این طرح تفاوت داشته باشند بسیاری از سیستم‌های محاوره‌ای، اجازه می‌دهند که کاربران از ترمینال، از پیش تایپ کنند یا قبل از درخواست داده، آنها را وارد کنند. در این حالت، وقفه می‌تواند رخ دهد که نشان دهنده رسیدن کرکرهای از ترمینال است، ولی در بلوک وضعیت-وسیله، انتظار پراسسی برای این وسیله ثبت نشده است. معمولاً، بافری در هر ترمینال وجود دارد.

مهمترین بهره I/O آسنکرون، افزایش بازدهی سیستم است. در حالی که I/O در حال انجام است، CPU سیستم می‌تواند پردازش یا آغاز کردن I/O، برای پراسس دیگری را اجرا نماید. از آن جایی که I/O نسبت به CPU سرعت پایین‌تری دارد، سیستم از امکاناتش استفاده بهتری می‌برد. در بخش ۲.۲.۲ مکانیزم دیگری برای بهبود کارآیی سیستم معرفی خواهیم کرد.

2.2.2. DMA Structure

DMA ساختمان

درایور ساده ورودی ترمینال را در نظر بگیرید. وقتی قرار است خطی از ترمینال خوانده شود، اولین کرکری که تایپ می‌شود، به کامپیوتر فرستاده می‌شود. وقتی آن کرکر دریافت می‌شود، دستگاه ارتباطی آسنکرون (یا پورت سریال) که خط ترمینال بدان وصل است، به CPU وقفه می‌فرستد. وقتی درخواست وقفه از ترمینال می‌رسد، CPU در حال اجرای دستورالعملی است. اگر در وسط سیکل اجرای دستورالعمل باشد، تا پایان اجرای دستورالعمل وقفه بر جا می‌ماند. سپس آدرس دستورالعمل بعدی ذخیره شده و کنترل به روشن سرویس دهنده وقفه، منتقل می‌شود.

1. idle

2. wait queue

3. ready-to-run

ویرف ای
 این روتین، رجیسترها لازم را ذخیره کرده و شرایط خطرا را تست می‌کند. آن گاه، کرکتر را از دستگاه گرفته و در بافری نگه می‌دارد و متغیرهای اشاره گر و شمارنده را تنظیم می‌نماید تا کرکتر بعدی در مکان بعدی بافر قرار گیرد. سپس، فلاگی^۱ را است می‌کند که به سایر بخش‌های سیستم عامل نشان می‌دهد که کرکتر جدید دریافت شده است. سایر قسمت‌ها مسئول پردازش کرکتر یا ارسال آن به برنامه مقاضی، می‌باشند (بخش ۵.۲ را بینید). سپس روتین سرویس (RTI) دهنده وقفه محتويات رجیسترها را بازیابی نموده و به دستور العمل وقفه دیده، باز می‌گردد.

اگر کرکترها در یک ترمینال^۲ باد^۳ تایپ شوند، زمان انتقال یک کرکتر توسط ترمینال ۱ میلی ثانیه یا ۱۰۰۰ مایکروثانیه می‌شود. یک روتین سرویس دهنده وقفه جهت وارد نمودن کرکترها در بافر، ممکن است به ۹۶۰۰ مایکروثانیه می‌شود. با فرض چنین اختلافی، به I/O آسنکرون، معمولاً حق تقدم پایینی نسبت داده دادن به سایر وقفه‌ها از سایر وسیله‌ها). با فرض چنین اختلافی، به I/O آسنکرون، معمولاً حق تقدم پایینی نسبت داده می‌شود و حتی قابل پس دادن نوبت به نفع وقفه دیگر هم می‌باشد. اما وسائل سریع، مانند دیسک، نوار، و شبکه‌های ارتباطی در سرعت قابل قیاس CPU کار می‌کنند؛ CPU به ۲ مایکروثانیه زمان برای پاسخ به هر وقفه نیاز دارد و وقفه‌ها هر ۴ مایکروثانیه (مثلاً) صادر می‌شوند. و زمان زیادی برای پردازش باقی نمی‌ماند.

برای حل این مشکلات، دسترسی مستقیم به حافظه^۴ (DMA) برای وسائل سرعت بالا استفاده می‌شود. بعد از تنظیم بافرها، اشاره گرها، و شمارندها برای وسائل I/O، دستگاه کنترل کننده وسیله، کل بلوک داده را مستقیماً از یابد، بافر خودش و حافظه بدون مداخله CPU، انتقال می‌دهد. تنها یک وقفه به ازاء هر بلوک صادر می‌شود به جای یک وقفه به ازاء هر کرکتر در وسائل سرعت پایین.

عمل اساسی CPU، بدین‌گونه است. برنامه کاربر یا خود سیستم عامل، ممکن است تقاضای انتقال داده نمایند. سیستم عامل یک بافر خالی برای ورودی یا یک بافر پر برای ارسال از انبار بافرها می‌جوید (برحسب نوع وسیله، بافر به طور نمونه ۱۲۸ یا ۲۰۴۸ بایت است). سپس بخشی از سیستم عامل به نام درایور وسیله^۵، رجیسترها کنترل کننده DMA را برای آدرس مناسب مبدأ و مقصد و طول بلوک انتقالی، سُت می‌نماید. آن گاه، کنترلر فرمان شروع انتقال را دریافت می‌کند و در این اثناء CPU آزاد است که سایر وظایف را اجرا نماید. از آن جایی که حافظه فقط یک کلمه^۶ حین انتقال DMA کنترل کننده DMA، سیکل‌هایی را از CPU می‌ذدد. این عمل سرعت کار CPU را در کنترل کننده، در پایان انتقال، کنترل کننده، به CPU وقفه می‌فرستد.*

3. direct memory access

* این روش cycle stealing نام دارد.

1. flag

2. baud

4. device driver

5. word

2.3. Storage Structure

۲.۲. ساختمان ذخیره

برنامه جهت اجرا بایستی در حافظه اصلی/قرار گیرد. حافظه اصلی تنها ناحیه ذخیره است که CPU می‌تواند مستقیماً دسترسی داشته باشد. حافظه برداری از بایت‌ها یا ورد هاست و اندازه مختلفی از ۱۰۰ ها هزار یا هزارها میلیون می‌تواند داشته باشد. هر ورد آدرس خودش را دارد. محاوره CPU و حافظه از طریق دستورالعمل‌های صریح load و store انجام می‌گیرد. علاوه بر آن، دستورالعمل‌ها، به طور اتوماتیک جهت اجرا از حافظه واکشی می‌شوند.

یک سیکل نمونه از اجرای دستورالعمل در کامپیوترهای فون نیومان^۱ بدین‌گونه است که ابتدا دستورالعمل از حافظه واکشی و در رجیستر دستورالعمل CPU قرار می‌گیرد، دیکود می‌شود و احتمالاً آدرس فیزیکی بر حسب مودهای آدرس دهی محاسبه و عملوند از حافظه به یکی از رجیسترهای کاری CPU خوانده می‌شود و عمل، انجام یافته و احتمالاً نتیجه در حافظه ذخیره می‌گردد. وقت کنید که واحد حافظه، جریانی از آدرس حافظه را می‌بیند. این که چگونه تولید می‌شوند (شمارنده برنامه، اندیس‌گذاری، غیر مستقیم، آدرس لیترال و غیره) یا چه هستند (دستورالعمل یا داده) را نمی‌داند. حسب الام، می‌توانیم نحوه تشکیل آدرس‌ها را چشم پوشی کرده و به توالی آدرس‌ها که توسط برنامه در حال اجرا، تولید می‌شود، توجه نماییم. حراز فری در ورنر^۲ مصیبرت را ممکن نمی‌سیند؟

به طورایده‌آل، مایل به ذخیره کل برنامه و داده به صورت دائم در حافظه اصلی هستیم که به دو دلیل ممکن نیست.

۱. حافظه اصلی زیادی کوچک است.

۲. حافظه اصلی، ذخیره میرا^۳ است و با قطع جریان برق، اطلاعات از دست می‌رود.

لذا اکثر سیستم‌های کامپیوتی، ذخیره ثانویه را به عنوان گسترش حافظه اصلی معرفی می‌کنند. معمول‌ترین وسیله ذخیره ثانویه، دیسک مغناطیسی است. اکثر برنامه‌ها (مورگرهای وب^۴، ویرایشگرهای کامپیوت، صفحات گسترده، و غیره) به طور ثابت بر روی دیسک قرار دارند تا جهت اجرا به حافظه اصلی، بار شوند. بنابراین، تمامی برنامه‌ها از دیسک به عنوان مبدأ و مقصد اطلاعات استفاده می‌کنند. لذا مدیریت دیسک در فصل ۱۲ مطالعه می‌شود.

اما، در مفهوم وسیع‌تر، ساختمان ذخیره‌ای که ذکر شد - شامل رجیسترها - حافظه اصلی و دیسک‌های مغناطیسی - بخشی از سیستم ذخیره ممکن هستند. حافظه‌های پنهانی^۵، سی‌دی رام‌ها^۶، نوارهای مغناطیسی و غیره هم وجود دارند. تفاوت‌های اصلی در میان سیستم‌های ذخیره متعدد مربوط به سرعت، هزینه، اندازه، و قابلیت میرایی آنهاست. در بخش‌های ۱۳.۲ و ۲۳.۲ در مورد حافظه اصلی، دیسک‌های مغناطیسی و نوارهای مغناطیسی که ادوات ذخیره مرسوم هستند، بحث می‌نماییم. در فصول ۱۲ و ۱۳ خواص مشخص بسیاری از وسائل ویژه، همانند

- 1. Von Neumann
- 4. cache memory

- 2. volatile
- 5. CD-ROM

- 3. web browser

دیسک‌های فلاپی، دیسک‌های هارد، CD-ROM‌ها و DVD‌ها بحث خواهند شد.

1.3.2. Main Memory

۱.۳.۲. حافظه اصلی

حافظه اصلی و رجیسترها دارای CPU هستند که مستقیماً توسط CPU دسترسی می‌شوند (بدین معنی که دستورالعمل‌ها، آدرس حافظه را به عنوان آرگومان می‌پذیرند ولی نه آدرس دیسک را). لذا، هر دستورالعمل در حال اجرا، یا هر دادهٔ مورد استفادهٔ دستورالعمل، بایستی در یکی از این وسائل ذخیرهٔ دستیابی مستقیم، قرار گیرند.

در مورد I/O، همان گونه که در بخش ۱.۲ ذکر شد، هر کنترل‌کننده‌ای، دارای رجیسترها‌یی جهت نگهداری فرمان و دادهٔ مورد انتقال می‌باشد. معمولاً، دستورالعمل‌های خاص I/O مجاز می‌دارند که داده از این رجیسترها به حافظه اصلی انتقال یابد. جیپت دسترسی راحت‌تر، اکثر معماری‌های کامپیوترا، I/O نگاشته شده^۱ در حافظه را مجاز می‌دارند. در این حالت، رنج‌هایی از حافظه کنار گذاشته شده و به رجیسترها و سایل، نگاشته می‌شوند. خواندن‌ها و نوشت‌ها به این آدرس‌های حافظه، سبب می‌شود داده‌ها به و از رجیسترها و سیله انتقال یابند. این روش برای نمایش، به یک مکان در حافظه اصلی نگاشته می‌شود. نمایش متن بر روی اسکرین، تقریباً به آسانی نوشت‌من در سکان‌های نگاشته شدهٔ حافظه مناسب می‌باشد.

I/O نگاشته شده برای سایر ادوات، مانند درگاه‌های سری و موازی که جهت اتصال مودم‌ها و چاپگرهای کامپیوترا به کار می‌روند، هم سهل است. CPU داده را از طریق این انواع وسائل با خواندن و نوشت‌من تعداد کمی رجیسترها و سیله به نام درگاه^۲ I/O، منتقل می‌کند. برای ارسال یک رشتهٔ طولانی از بایت‌ها از طریق پورت سری نگاشته شدهٔ حافظه، CPU یک بایت داده را در رجیستر داده می‌نویسد، سپس بیتی را در رجیستر کنترل می‌کند و سپس سیگنال می‌دهد که داده آماده است. وسیله، بایت داده را گرفته، و سپس بیت رجیستر کنترل را پاک می‌کند تا نشان دهد که آماده دریافت بایت بعدی است. آن‌گاه CPU می‌تواند بایت دیگری را بفرستد. اگر CPU برای مشاهده بیت کنترل، پولینگ را به کار برد، روش عمل را I/O برنامه شده^۳ (PIO) می‌نامند. اگر CPU درگاه کنترل را بازرسی ننماید، بلکه دستگاه، اعلام وفته برای آمادگی قبول داده بعدی، نماید، انتقال داده، رانده شده با وفته^۴ نامیده شود.

رجیسترها‌یی که درون CPU ساخته می‌شوند، عموماً در یک سیکل ساعت CPU قابل دسترسی‌اند. و چندین عمل رجوع به این رجیسترها در یک تیک ساعت اجرا می‌شوند. ولی در مورد حافظه اصلی چنین نیست. دستیابی از

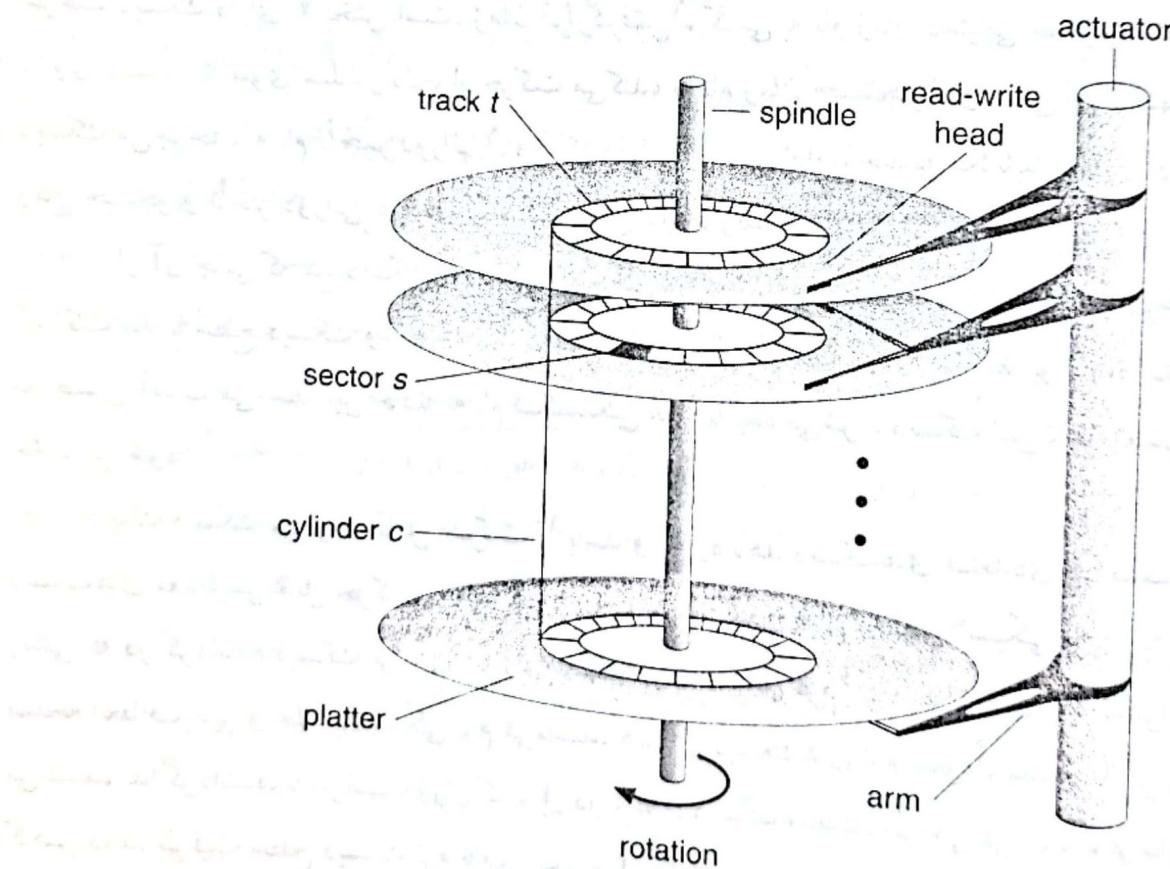
- 1. memory-mapped I/O
- 2. I/O port
- 3. programmed I/O
- 4. interrupt driven

طریق تراکنش^۱ بر روی گذرگاه حافظه انجام می‌گیرد. این عمل، نیازمند چندین سیکل ساعت است و لذا CPU لازم است که کنترل^۲ شود زیرا داده آمده را ندارد. این وضعیت، به علت مراجعات مکرر به حافظه، غیرقابل تحمل است. چاره^۳ آن است که حافظه سریعی مابین CPU و حافظه اصلی اضافه شود. بافر حافظه به کار رفته در این تفاصل سرعت، حافظه نهانی^۴ نامیده می‌شود و در بخش ۱۰.۴.۲ شرح داده شده است.

2.3.2. Magnetic Disks

۲.۳.۲. دیسک‌های مغناطیسی

دیسک‌های مغناطیسی، انبوه^۵ ذخیره ثانویه را برای سیستم‌های کامپیوتری مدرن، فراهم می‌آورند) از جث مفہوم، دیسک‌ها نسبتاً ساده هستند (شکل ۲-۵). هر صفحه^۶ دیسک دارای شکل مدور مسطح، مانند یک



شکل ۲-۵: مکانیزم دیسک با هد متحرک

- 1. transaction
- 2. to stall
- 4. cache memory
- 5. bulk

- 3. remedy
- 6. platter

سی باشد. قطر صفحات معمول، از ۱/۸ تا ۵/۲۵ اینچ در تغییر است. هر دو سطح صفحه از ماده مغناطیسی نظری نوارهای مغناطیسی پوشانده شده است. ما اطلاعات را با ثبت مغناطیسی وار بر روی دیسک نگهداری می‌کنیم.^۱ نوارهای مغناطیسی پوشانده شده است. تمام هدایت‌های را با ثبت مغناطیسی وار بر روی دیسک نگهداری می‌کنیم.^۲ هدایت‌های را با ثبت مغناطیسی پوشانده شده است. هر سطح صفحه به طور منطقی به دوایر متحدد مرکزی به نام شیار^۳ تقسیم شده است که آن خود به قطاع‌هایی^۴ تقسیم می‌شود. مجموعه شیارهایی که در یک موقعیت بازو قرار دارند و هم شعاعده، سیلندر^۵ نام دارد. در یک گردانده دیسک می‌تواند هزاران سیلندر متحدد مرکز قرار گیرد و هر شیار صدها سکتور را می‌تواند دارا باشد. ظرفیت ذخیره دیسک‌های معمول بر حسب گیگابایت است (۳۰^۶ بایت، تقریباً ۱ بیلیون بایت).

زمانی که دیسک در حال استفاده است، یک موتور گردانده، آن را با سرعت بالا می‌چرخاند. اکثر گردانده‌ها با سرعت ۶۰ تا ۱۵۰ دور در ثانیه می‌چرخند. میزان انتقال^۷، میزان جریان داده مابین کامپیوتر و گردانده است. سرعت دیسک دارای ۲ بخش است. زمان قرارگرفتن^۸، گاهی به نام زمان دستیابی مستقیم، شامل زمانی است که بازوی دیسک به سوی سیلندر دلخواه حرکت می‌کند، به نام زمان جستجو^۹، و زمانی که سکتور مطلوب به زیر هد دیسک می‌چرخد، به نام تأخیر دورانی^{۱۰}. دیسک‌های نمونه، می‌توانند چندین مگابایت داده را در ثانیه انتقال دهند، و زمان جستجو و تأخیر دورانی در حدود چندین میلی ثانیه دارند.

از آن جایی که هد دیسک در فضای کوچکی در هوا بر روی دیسک پرواز می‌کند، (بر حسب میکرون) خطر کشات هد با سطح دیسک وجود دارد. اگرچه سطح دیسک‌ها با لایه ماده محافظ پوشانده شده است، اما به سطح مغناطیسی آسیب می‌رسد. این حادثه به نام شکستگی هد^{۱۱} نامیده می‌شود. دیسک نمی‌تواند تعمیر شود و بایستی تماماً جایگزین شود.

یک دیسک می‌تواند قابل حرکت^{۱۲} باشد و اجازه دهد دیسک‌های متعددی بر حسب نیاز سوار شوند. دیسک‌های مغناطیسی قابل حرکت عموماً دارای یک صفحه‌اند و دارای جلد پلاستیکی برای پیشگیری از خرابی در زمانی که در گردانده دیسک قرار ندارند، می‌باشند. دیسک‌های نرم^{۱۳}، دیسک‌هایی مغناطیسی قابل حمل و دارای صفحه اعطاف‌پذیر و جلد پلاستیکی نرم می‌باشند. هد‌های دیسک‌های نرم عموماً، مستقیماً بر روی سطح دیسک می‌نشینند. لذا اگر گردانده، با سرعت دوران کمتر از درایوهای دیسک سخت، طراحی می‌شود تا فرسایش سطح دیسک را کاهش دهد. ظرفیت سطح دیسک نرم به طور نمونه ۱ مگابایت یا بیشتر می‌باشد. دیسک‌های قابل حرکتی وجود دارند

- | | | |
|--------------|-----------------------|---------------------|
| 1. disk-arm | 2. track | 3. sector |
| 4. cylinder | 5. transfer rate | 6. positioning time |
| 7. seek time | 8. rotational latency | 9. head crash |
| 10. movable | 11. floppy | |

- | |
|---------------------|
| 3. sector |
| 6. positioning time |
| 9. head crash |

که بیشتر شبیه دیسک سخت نرمال کار می‌کنند، و ظرفیت در مقیاس گیگابایت دارند. یک گرداننده دیسک، توسط سیم‌هایی به نام گذرگاه I/O به کامپیوتر متصل می‌شود. انواع متعددی از گذرگاه‌ها در دسترسند، شامل SCSI، EIDE و میزبان^۱. انتقالات داده بر روی یک گذرگاه، توسط پردازشگرهای الکترونیکی ویژه، به نام کنترلرهای^۲ انجام می‌گیرند. کنترلرهای میزبان در انتهای کامپیوتری باس قرار دارد. یک کنترلر دیسک^۳ در هر گرداننده دیسک ساخته می‌شود. جهت انجام یک عمل I/O دیسک، کامپیوتر فرمانی در کنترل کننده میزبان می‌گذارد، به طور نمونه با استفاده از درگاه‌های نگاشته در حافظه، همان‌گونه که در بخش ۲.۳.۱ شرح داده شد. کنترل کننده میزبان، سپس، فرمان را از طریق^۴ پیغام‌ها به کنترل کننده دیسک می‌فرستد، و این کنترل کننده، سخت‌افزار درایو دیسک را به کار می‌اندازد. کنترل کننده‌های دیسک معمولاً یک حافظه کش تو-ساخته^۵ دارند. انتقال داده در درایو دیسک، مابین کش و سطح دیسک انجام می‌گیرد و انتقال داده به میزبان، در سرعت‌های الکترونیکی سریع، مابین کش و کنترل کننده میزبان رخ می‌دهد. مراحل^۶ از پیش‌نگهداشت تا اینجا در اینجا^۷ مذکور

2.3.3. Magnetic Tapes

۲.۳.۲. نوارهای مغناطیسی

نوارهای مغناطیسی^۶ به عنوان رسانه ذخیره ثانویه قدیمی مورد استفاده قرار می‌گرفتند. اگرچه آن‌ها نسبتاً دایمی هستند، و می‌توانند مقادیر زیادی داده را نگه دارند اما، زمان دستیابی شان در مقایسه با حافظه اصلی، بسیار کند است. دسترسی تصادفی^۷ به نوارهای مغناطیسی هزاران بار کنترل از دیسک‌های مغناطیسی است، لذا امروزه رسانه ثانویه مناسبی نیستند. نوارها، اساساً به عنوان پشتیبان برای روز مبادا، استفاده دارند و برای نگهداری اطلاعات به ندرت مورد استفاده در آرشیو و نیز به عنوان وسیله‌ای برای انتقال اطلاعات از یک کامپیوتر به دیگری استفاده می‌شوند. نوار در یک ریل قرقه^۸ نگهداری می‌شوند. و از مقابل یک هد خواندن و نوشتن می‌چرخد.^۹ حرکت به یک نقطه درست در روی نوار، می‌تواند دقایقی به طول انجامد، اما همین که پیدا شد، درایوهای نوار می‌توانند در سرعت قابل مقایسه با درایوهای دیسک، داده‌ها را بنویسن. ظرفیت نوارها در رنج وسیعی تغییر می‌کند و بستگی به درایو نوار ویژه، دارد. بعضی از نوارها ۲۰ برابر بیشتر از یک درایو دیسک سخت، توانایی ذخیره اطلاعات را دارند. نوارها برحسب عرض شامل ۴، ۸ و ۱۶ میلی متر و $\frac{1}{2}$ و $\frac{1}{4}$ اینچ طبقه‌بندی می‌شوند.

نمایه نوار مغناطیسی^{۱۰} از فرآیندر قدرمی^{۱۱} نتیج برعضطری از اوزنبارا^{۱۲} صور راسته^{۱۳} در آرایه^{۱۴} مهاد راز^{۱۵} را در اینجا^{۱۶} اینجا^{۱۷} اطلاعات^{۱۸} را که مسوز^{۱۹} در^{۲۰} از

۱. controller

۴. via

۷. random access

2. host controller

5. built-in cache

8. spool

3. disk controller

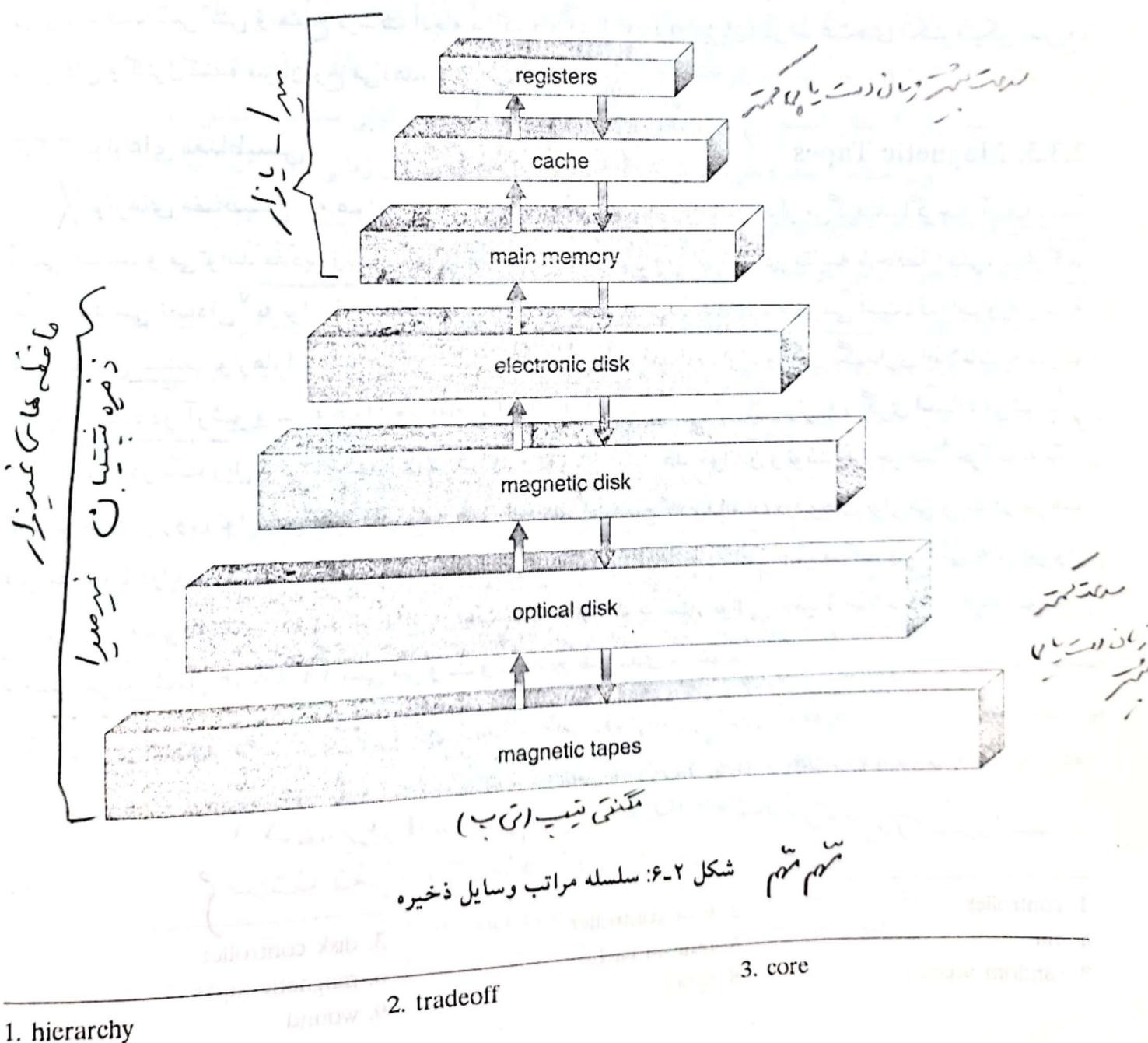
6. magnetic tapes

9. wound

2.4. Storage Hierarchy

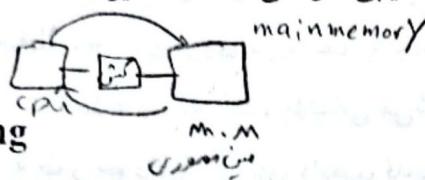
۴.۲. سلسله مراتب ذخیره

در یک سیستم کامپیووتری، انواع وسیعی از سیستم‌های ذخیره می‌توانند بر حسب سرعت و قیمت شان در یک سلسله مراتب، سازماندهی شوند (شکل ۲-۶). سطوح بالاتر گران، اما سریع هستند. هرچه که از سلسله مراتب پایین می‌رویم، هزینه به ازاء بیت، عموماً کاهش می‌یابد، در حالی که زمان دستیابی عموماً افزایش می‌یابد^۱ این، تعادل^۲ مستدل است. در واقع، بسیاری از ادوات قدیمی، شامل نوار کاغذی و حافظه‌های کور^۳، اکنون در موزه‌ها دیده می‌شوند زیرا که نوارهای مغناطیسی و حافظه‌های نیمه‌هادی، سریع‌تر و ارزان‌تر شده‌اند. هرچه قدر بالاتر در درجه ارزانی و افزایش در سرعت کاهش زبان دست یافته خواهیم داشت.



شکل ۲-۶: سلسله مراتب وسائل ذخیره

علاوه بر قیمت و سرعت سیستم‌های ذخیره‌گوناگون، موضوع میرایی^۱ ذخیره نیز مطرح است ~~ذخیره میرا~~
محتویات خود را در اثر قطع نیروی الکتریکی از دست می‌دهد ~~در غیاب سیستم‌های یدک دارای ژنراتور و یا باطری~~
گران‌قیمت، داده بایستی در محیط ذخیره غیر میرا به منظور نگهداری امن، نوشته شود. در سلسله مراتب نشان داده شده
سیستم‌های ذخیره بالای دیسک‌های گوناگون، میرا هستند در حالی که، سیستم‌های زیر حافظه اصلی غیرمیرا^۲ می‌باشند.
طراحی سیستم حافظه کامل بایستی تمامی این فاکتورها را متعادل سازد: یعنی تنها آن مقدار حافظه گران‌قیمت را که
ضروری است، به کار برد و در عین حال به قدر ممکن از حافظه ارزان‌تر و غیرمیرا، بیشتر استفاده نماید. حافظه‌های
نهانی برای تعادل تفاوت‌های کارآیی، در مواقعي که زمان-دستیابی طولانی و تفاوت میزان انتقال مایین دو مؤلفه زیاد
است، تعییه می‌شوند.



2.4.1. Caching

۲.۴.۲. نهانی کردن بسیار صعب

یکی از اصول مهم سیستم‌های کامپیوتری، نهانی کردن^۳ است. اطلاعات، به طور طبیعی، در سیستم ذخیره‌ای
نگه داشته شده است (مانند حافظه اصلی). در اثنايی که مورداستفاده قرار می‌گیرد، به طور موقت، به یک سیستم ذخیره
سریع‌تر - کش - کپی می‌شود. زمانی که یک تکه ویژه اطلاعات موردニاز است، ابتدا به حافظه نهانی رجوع می‌کنیم.
چنان‌چه در آن موجود باشد، مستقیماً اطلاعات را از کش می‌خوانیم؛ در غیر این صورت، اطلاعات از حافظه اصلی
خوانده می‌شود و با فرض احتمال بالا مبنی بر نیاز آتی دوباره، یک کپی از آن در حافظه کش قرار داده می‌شود ~~X~~
با تعمیم دید فوق، رجیسترهای برنامه‌پذیر داخلی، مانند رجیسترهای شاخص^۴، یک کش سریع برای حافظه
اصلی فراهم می‌سازند. برنامه نویس (یا کامپایلر)، الگوریتم‌های تخصیص-رجیستر و جایگزینی-رجیستر را جهت
تفصیل گیری این که کدام داده در رجیسترها و کدام در حافظه اصلی قرار گیرد، پیاده‌سازی می‌کنند. کش‌هایی نیز وجود
دارند که به‌تمامی در سخت‌افزار پیاده‌سازی می‌شوند. به عنوان مثال، اکثر سیستم‌ها، دارای کشِ دستورالعمل^۵ هستند،
که دستورالعمل‌های بعدی موردنظر جهت واکشی^۶ را در خود جای می‌دهند ~~بدون این کش، CPU~~ مجبور است
سیکل‌های متعددی در حین واکشی دستورالعمل از حافظه اصلی^۷ منتظر بمانند. بنابراین مشابه، اکثر سیستم‌ها، در
سلسله مراتب حافظه، یک یا بیشتر کش داده^۸، دارند ~~در این متن، به این گونه کش‌های فقط سخت‌افزاری، مربوط~~
نمی‌شوند زیرا آنها خارج از محدوده کنترل سیستم عامل قرار دارند.

از آنجایی که کش‌ها اندازه محدودی دارند، مدیریت نهانی^۹، مسئله طراحی مهمی است. انتخاب دقیق اندازه
کش و سیاست جایگزینی^۹، سبب می‌گردد که ۸۰ تا ۹۹ درصد کل دستیابی‌ها در کش یافته شود و لذا بازدهی

1. volatility

2. non-volatile

3. caching

4. index

5. instruction cache

6. fetch

7. data cache

8. cache management

9. replacement policy

نحوه اول با main memory اینکه این را در کس کش می‌کنیم و کد مرتبه این کش را در زیر قرار دهیم ~~و در کس کش~~
آخر کش را در داریم. این را در کس کش می‌دانیم از آنکه؟ مفاهیمی که پیش از اینکه این را در کس کش داریم، و با مفاهیمی که پیش از اینکه این را در کس کش داریم.

فوق العاده بالا حاصل گردد\ الگوریتم‌های جایگزینی متعددی برای کش‌های کنترل شده - نرم‌افزار در فصل ۹ بحث خواهند شد.

حافظه اصلی می‌تواند نظیر یک کش سریع برای حافظه ثانویه باشد، زیرا داده موجود در فضای ذخیره ثانویه برای استفاده باید به حافظه اصلی کپی شود و داده بایستی قبل از ذخیره در حافظه ثانویه به منظور ذخیره امن (دائم)، در حافظه اصلی قرار داشته باشد. داده سیستم - فایل می‌تواند در سطوح مختلف سلسله مراتب ذخیره ظاهر شود. در بالاترین سطح، سیستم عامل می‌تواند یک کش در حافظه اصلی برای داده سیستم - فایل فراهم آورد. همین طور، دیسک‌های الکترونیکی RAM^۱ می‌توانند در حافظه‌های سریع میرا جهت دسترسی از طریق واسطه سیستم - فایل، به کار روند. انبوه حافظه ثانویه، بر روی دیسک مغناطیسی است. ذخیره دیسک مغناطیسی هم گاهی اوقات توسط نوارهای مغناطیسی و دیسک‌های متحرک، پشتیبانی می‌شود تا در اثر خرابی دیسک سخت، داده‌ها محافظت شوند. بعضی از سیستم‌ها، به طور خودکار فایل‌های داده‌ای قدیمی را از ذخیره ثانی به ذخیره ثالث^۲ نظیر ژوک باکس‌های نواری^۳ آرشیو می‌کنند (بخش ۳.۰.۱۳ را ببینید).

حرکت اطلاعات در سطح تسلسلی ذخیره، می‌تواند صریح یا ضمنی باشد که بستگی به طراحی سخت‌افزار و نرم‌افزار سیستم عامل کنترل کننده، دارد. به عنوان مثال، انتقال داده از کش به CPU و ثبات‌ها^۴ معمولاً یک تابع سخت‌افزاری است و سیستم عامل دخالتی ندارد. از سوی دیگر، انتقال داده از دیسک به حافظه معمولاً توسط سیستم عامل کنترل می‌شود.

ناتایج، inconsistency

۲.۴.۲. Coherency and Consistency

در یک ساختمان ذخیره تسلسلی، داده یکسانی می‌تواند در سطوح مختلف سیستم ذخیره وجود داشته باشد. به عنوان مثال، فرض کنید متغیر صحیح A در فایل B واقع شده و باید یک واحد افزایش یابد. فرض کنید فایل B بر روی دیسک مغناطیسی قرار دارد. عمل افزایش، ابتدا با صدور عمل I/O، بلوک دارای A را از فایل B به حافظه اصلی می‌آورد. عمل مذکور احتمالاً با کپی A در کش و نیز کپی A در یک رجیستر داخلی، ادامه می‌یابد. بنابراین کپی A در محل‌های متعددی واقع است. همین که افزایش در رجیستر داخلی انجام گیرد، مقدار A در سیستم‌های ذخیره متعدد، متفاوت می‌شود. این مقدار تنها زمانی که مقدار جدید A در دیسک مغناطیسی بازنویسی شود، یکسان خواهد شد.

در محیطی که تنها یک پردازش در یک لحظه در حال اجراست، این تشکیلات هیچ مشکلی ندارد، زیرا دسترسی به متغیر صحیح A همواره در بالاترین سطح تسلسلی، خواهد بود^۵ اما، در محیط چند وظیفه‌گی، که پرازنده مابین چندین پردازش پس و پیش سوئیچ می‌کند، نهایت دقت بایستی به کار رود تا تضمین شود که، اگر پراسس‌های

3. tape jukeboxes

2. tertiary storage

1. RAM disks

4. registers

متعددی مایل به دسترسی به A هستند، آن گاه هریک از آنها مقدار به روز درآمده اخیر ترین را به دست خواهد آورد. وضعیت، در محیط چند-پردازنده‌ای، پیچیده‌تر می‌گردد. در آن جا علاوه بر نگهداری رجیسترهاي داخلی، CPU دارای کش محلی نیز هست. در یک چنین محیطی، یک کپی از A ممکن است همزمان در چندین کش وجود داشته باشد. از آن جایی که، CPU‌های متعددی همروند اجرا می‌شوند، بایستی اطمینان دهیم که به روز در آوردن مقدار A در یک کش، فوراً به تمام کش‌های بقیه که A در آنها قرار دارد، منعکس می‌شود. این مسئله به نام انسجام کش نامیده می‌شود، و عموماً یک موضوع سخت‌افزاری است (در زیر سطح سیستم عامل انجام می‌گیرد). در یک محیط توزیع شده، وضعیت حتی پیچیده‌تر می‌گردد. در یک چنین محیطی، کپی‌های متعددی (نسخه‌های مکرر^۱) از یک فایل یکسان می‌توانند در کامپیوتراهای مختلف که در فضا گسترشده‌اند، نگهداری شوند. از آن جایی که نسخه‌های متعدد می‌توانند به طور همروند دستیابی شده و به هنگام آورده شوند، بایستی مطمئن شویم که، وقتی نسخه‌ای در یک محل به روز آورده می‌شود، آن گاه کلیه نسخه‌های دیگر هرچه سریعتر، بهنگام آورده خواهند شد.

2.5. Hardware Protection

۵.۲ حفاظت سخت‌افزاری

سیستم‌های کامپیوتری قدیمی، سیستم‌های تک-کاربره‌ای بودند که توسط برنامه‌نویس به کار می‌افتادند. وقتی برنامه‌نویسان کامپیوتر را از کنسول به کار می‌انداختند، کنترل کامل بر روی سیستم داشتند. در اثناء تکامل سیستم‌های عامل، این کنترل به آنها واگذار شد. با ظهور برنامه‌مانیتور^۲ مقیم، سیستم عامل شروع به اجرای بسیاری از وظایف نمود، به ویژه I/O، که قبلًاً توسط برنامه‌نویس انجام می‌گرفت.

علاوه بر افزایش بهره‌وری سیستم، سیستم عامل شروع به تقسیم منابع سیستم مابین برنامه‌های گوناگون به طور همزمان کرد. با اسپولینگ، یک برنامه می‌توانست در حال اجراشدن باشد در حالی که برنامه‌های دیگر عمل I/O انجام می‌دادند. دیسک همزمان داده پراسس‌های بسیاری را نگه می‌داشت. چند برنامگی چندین برنامه را در یک زمان در حافظه می‌گذاشت.

این اشتراک، هم بهره‌وری بهبود یافته و هم مشکلات بیشتر را ایجاد نمود. وقتی سیستمی بدون اشتراک اجرا می‌شد، خطای یک برنامه فقط در آن مؤثر بود. در اشتراک گذاردن، پراسس‌های زیادی، می‌توانند از خطای موجود در یک برنامه، متأثر شوند.

به عنوان مثال، سیستم عامل ساده دسته‌ای را در نظر بگیرید (بخش ۲.۱)، که چیزی بیش از توالی اتوماتیک کارها، ارائه نمی‌دهد. فرض کنید برنامه‌ای در لوب خواندن کارت‌های ورودی گیر کند. برنامه کل داده خود را می‌خواند و اگر چیزی او را متوقف نکند، خواندن کارت‌های داده کار بعدی و بعدی و الی آخر را ادامه خواهد داد.

1. replica

2. resident monitor

این حلقه، عمل درست کارهای بسیاری را جلوگیری خواهد نمود. حتی خطاهای پیچیده‌تری در سیستم مالتی پروگرامینگ می‌توانند رخ دهند. جایی که یک برنامه غلط می‌تواند برنامه یا داده برنامه دیگری را تغییر دهد یا حتی خود برنامه مانیتور مقیم را عوض کند. Macintosh OS، MS-DOS هردو این نوع خطأ را مجاز می‌دارند.

بدون محافظت از اینگونه خطاهای، یا کامپیوتر باستی هر بار یک برنامه را اجرا کند، یا تمام خروجی باید مشکوک باشد. یک سیستم عامل درست طراحی شده، باستی اطمینان دهد که یک برنامه نادرست (یا بدخواه) نمی‌تواند سبب شود که سایر برنامه‌ها نادرست اجرا شوند.

بسیاری از خطاهای برنامه سازی، توسط سخت‌افزار کشف می‌شوند. این خطاهای عموماً توسط سیستم عامل بررسی می‌شوند. اگر برنامه کاربری به دلیلی، مثلاً سعی در اجرای دستور العمل غیرقانونی، یا دسترسی حافظه‌ای که در فضای آدرس کاربر نیست، رد شود، آن‌گاه سخت‌افزار به سیستم عامل تله^۱ می‌شود. تله، کنترل را از طریق بردار و قوه به سیستم عامل انتقال می‌دهد. هرگاه، یک خطای برنامه رخ دهد، سیستم عامل به طور غیرطبیعی برنامه را متوقف می‌نماید، پیغام خطای مناسب فرستاده می‌شود، و حافظه برنامه رونوشت می‌شود و معمولاً بر یک فایل نوشته می‌شود تا کاربر بتواند آن را تست کرده و شاید تصحیح نموده و مجددآ اجرا نماید.

mod ریزکر (ردپریز)

2.5.1. Dual-Mode Operation

۲.۵.۲. عمل وجه - دوگانه کاربری mode

برای تضمین عمل درست، باستی سیستم عامل و سایر برنامه‌ها و داده‌هایشان را از برنامه بدکار محافظت نمایم. حفاظت برای هر منبع تقسیم شده ضروری است. روش به کار رفته، این است که حمایت سخت‌افزاری ای فراهم نمود تا بتوان مابین حالت‌های مختلف اجرا تفاوت قائل شد~~ل~~ در خیلی پایین ترین سطح، به دو حالت جداگانه نیازمندیم: حالت کاربر^۲. حالت مانیتور^۳ (که مود سرپرست^۴، مود سیستم، یا مود ممتاز^۵ هم نامیده می‌شود). یک بیت، به نام بیت مود^۶ به سخت‌افزار اضافه می‌شود تا مود کنونی را نشان دهد. با این بیت، می‌توانیم تشخیص دهیم که اجرا به نام سیستم عامل است و یا به نام کاربر. همان‌گونه که خواهیم دید، این تمامیت معماً معماری - گونه برای بسیاری از وجوده عمل سیستم عامل مفید است.

در زمان بوت سیستم، سخت‌افزار در مود مانیتور آغاز می‌شود. سپس سیستم عامل بار می‌شود، و پردازش‌های کاربر را در مود کاربر اجرا می‌کند. هر زمان که تله یا وقفه‌ای رخ داد، سخت‌افزار از مود کاربر به مود مانیتور همچیج می‌کند (مثلاً بیت مود از ۱ به ۰ تغییر می‌کند). لذا، هرگاه که سیستم عامل کنترل کامپیوتر را به دست گیرد، در مود مانیتور است. قبل از عبور به برنامه کاربر، سیستم به مود کاربر سوئیچ می‌کند. عمل مود-دوگانه، وسیله‌ای جهت

1. trap

4. supervisor mode

2. user mode

5. privileged mode

3. monitor mode

6. bit mode

Mod منیتور بدان تکریان درستس: سیستم را از این راه نمایه بردار.

mod ماست بار.

حفظه سیستم عامل از کاربر خطأکار و کاربران خطأکار از یکدیگر فراهم می‌آورد. این محافظت بدین گونه صورت می‌گیرد که بعضی از دستورالعمل‌های ماشین که باعث زیان می‌شوند، به مانند دستورات ممتاز^۱ طراحی می‌گردند. این دستورات فقط در مود مانیتور قابل اجرا هستند. اگر در مود کاربر به کار روند سخت‌افزار آنها را اجرا نکرده‌فر ^{رـسـخـنـدـ} ~~نـیـتـورـسوـئـیـهـ~~ ^{نـیـتـورـسوـئـیـهـ} ^{نـیـتـورـسوـئـیـهـ} ^{نـیـتـورـسوـئـیـهـ} ^{نـیـتـورـسوـئـیـهـ} ^{نـیـتـورـسوـئـیـهـ} ^{نـیـتـورـسوـئـیـهـ} غیرقانونی^۲ تشخیص داده و تلـهـسـیـسـ عـاـمـل مـیـشـونـد.

کسبود مود-دوگانه تحت حمایت سخت‌افزار، می‌تواند زیان جدی در سیستم عامل ایجاد نماید. به عنوان مثال MS-DOS برای معما رای اینتل ۸۰۸۸ نوشته شده است، که قادر بیت مود و لذا مود-دوگانه می‌باشد. یک برنامه کاربر با اجرای غلط می‌تواند با رونویسی بر داده سیستم عامل، آن را پاک کند. برنامه‌های متعدد قادرند در یک زمان در وسیله I/O بنویسند و احتمالاً نتایج فاجعه‌آمیز پدید آورند. ورژن‌های اخیر و پیشرفته‌تر CPU‌های اینتل، نظیر ۸۰۴۸۶، در مود-دوگانه عمل می‌کنند. و در نتیجه، سیستم‌های عامل اخیر، مانند مایکروسافت ویندوز NT و IBM OS/2 از این ویژگی بهره برده و محافظت گسترده‌تری برای سیستم عامل فراهم می‌سازند.

2.5.2. I/O Protection

I/O ۲.۵.۲

یک برنامه کاربر، می‌تواند با صدور دستورالعمل‌های I/O غیرقانونی، با دستیابی به مکان‌های حافظه سیستم عامل، یا با عدم رهایی CPU، کار طبیعی سیستم را مختل نماید. مکانیزم‌های متعددی می‌توان به کار برد تا تضمین شود چنین اختلالی در سیستم رخ نمی‌دهد.

برای پیشگیری کاربری از اجرای I/O غیرقانونی، کلیه دستورالعمل‌های I/O را به صورت دستورالعمل‌های ممتاز، تعریف می‌کنیم. لذا کاربران دستورات I/O را مستقیماً صادر نمی‌کنند؛ باستی آنها را از طریق سیستم عامل انجام دهند. برای حفاظت کامل I/O باستی مطمئن شویم که برنامه کاربر در مود مانیتور، کنترل سیستم کامپیوتر را به دست

نمی‌آورد.

کامپیوتری را که در مود کاربر کار می‌کند، درنظر بگیرید. هر زمان که وقفه یا تله رخ دهد، به مود مانیتور سوئیچ می‌شود و به آدرسی در بردار وقفه، پرش انجام می‌گیرد. فرض کنید که برنامه کاربر، آدرسی در بردار وقفه را، به عنوان بخشی از اجرای خود، عوض نماید در این صورت هرگاه که وقفه رخ دهد، سخت‌افزار به مود مانیتور سوئیچ نموده و کنترل به این آدرس تغییر یافته در بردار وقفه، متعلق به فضای آدرس کاربر، منتقل می‌شود! برنامه کاربر کنترل سیستم کامپیوتر را در مود مانیتور به دست آورده است.

2.5.3. Memory Protection

۳.۵.۲ حفاظت حافظه

جهت تضمین عمل صحیح، باستی بردار وقفه را از تغییر یافتن توسط برنامه کاربر، محافظت نماییم. به علاوه،

1. privileged instructions

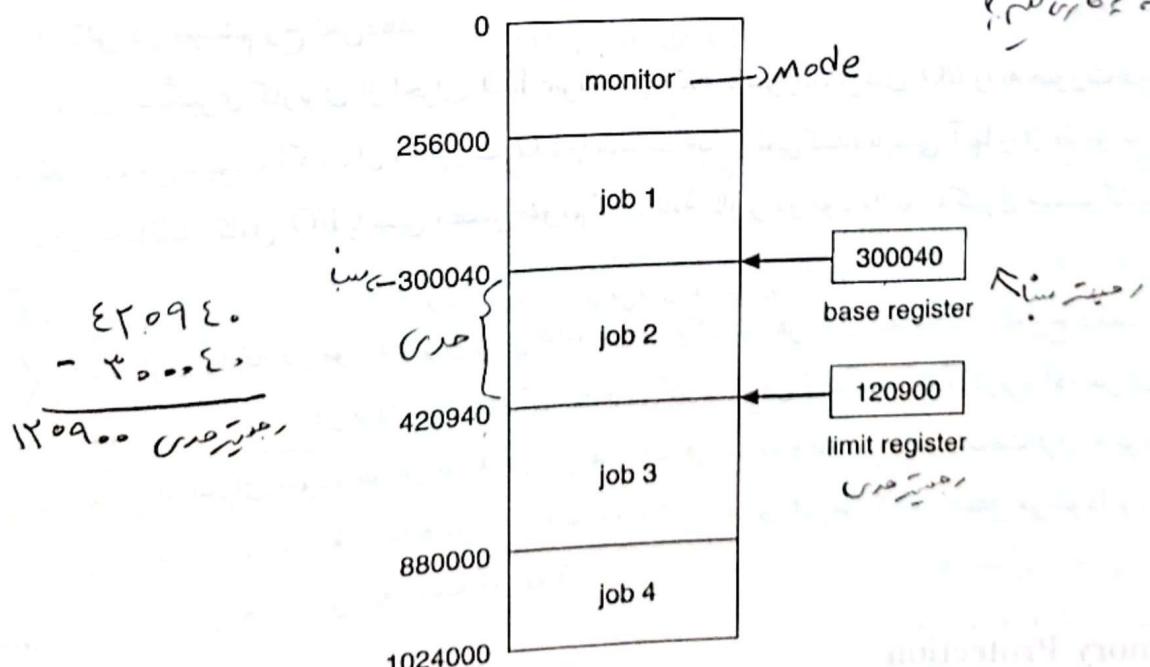
2. illegal

روال‌های سرویس دهنده و قله در سیستم عامل را نیز باید حفاظت کنیم در غیر این صورت برنامه کاربر ممکن است دستورات این روتین را دوباره نویسی نموده و با قرار دادن دستور پرش به برنامه، از روتین سرویس دهنده و قله به مود مانیتور دسترسی پیدا کند. حتی اگر کاربر، کنترل غیر مجاز به دست نیاورد، تغییر روتین و قله سبب مخدوش شدن کار درست سیستم و اسپولینگ و بافرگیری می‌گردد.

مشاهده می‌کنیم که حداقل بایستی برای بردار و قله و روال‌های سرویس دهنده و قله، محافظت حافظه صورت گیرد. اما، در حالت کلی، می‌خواهیم سیستم عامل را از دسترس برنامه‌های کاربر و نیز، برنامه‌های کاربر را از یکدیگر، در امان نگه‌داریم. این حفاظت می‌باشد توسط سخت‌افزار صورت گیرد. این کار به شیوه‌های گوناگونی در فصل ۸ پیاده‌سازی خواهد شد. در اینجا یک پیاده‌سازی ممکن، به اجمالی ارائه می‌شود.

آن‌چه که نیاز داریم تا فضای حافظه برنامه‌ای را جدا سازد، توانایی تشخیص رنج آدرس‌های مجاز قابل دسترسی برنامه است تا سایر بخش‌های حافظه خارج این فضا، حفاظت شوند^۱ با استفاده از دو رجیستر حدی^۱ و مبنای^۲ مطابق (شکل ۷-۲) این عمل صورت گیرد. رجیستر مبنای، کوچک‌ترین آدرس فیزیکی قانونی را شامل است؛ و رجیستر حدی، اندازه محدود را در بر می‌گیرد^۳ مثلاً مطابق شکل برنامه رنج آدرس ۴۰۰۰۴۰ - ۳۰۰۰۴۰ را دستیابی خواهد نمود.

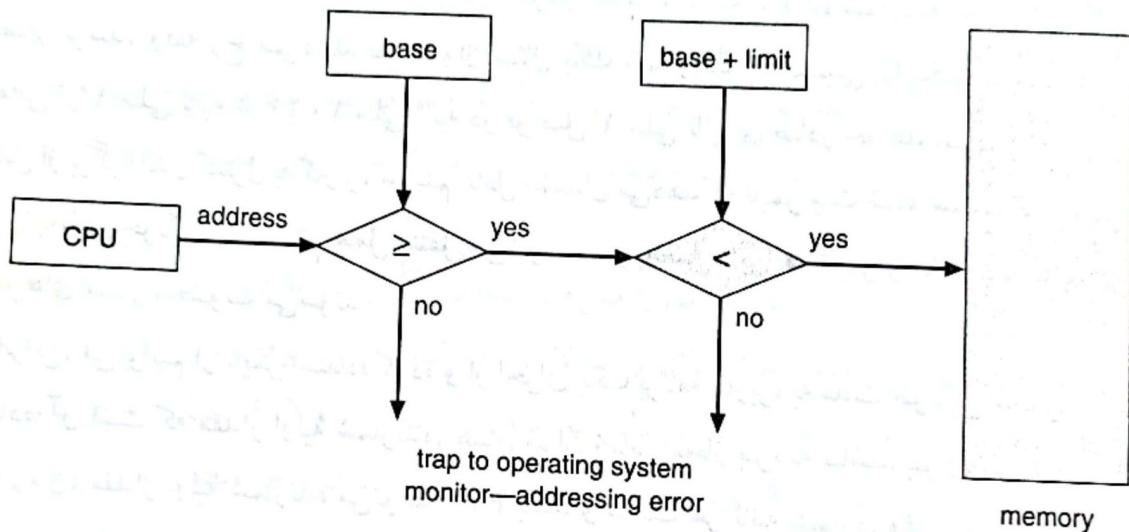
روال ایمان



شکل ۷-۲: رجیسترهاي حدی و مبنای فضای آدرس منطقی را تعریف می‌کنند.

- 1. limit register
- 2. base register

این حفاظت توسط سخت‌افزار CPU با مقایسه هر آدرس تولید شده در مود کاربر با رجیسترها صورت می‌گیرد. هر کوششی در دسترسی به سیستم عامل یا فضای آدرس برنامه‌های دیگر، سبب تله به مانیتور می‌شود و این کار خطای مهلک^۱ محسوب می‌شود (شکل ۲-۸). این طرح، برنامه‌کاربر را از تغییر دادن (اتفاقی یا عمدی) کد یا داده سیستم عامل و سایر کاربران منع، می‌نماید.



شکل ۲-۸: حفاظت آدرس سخت‌افزاری با ثبات‌های حدّی و مبنا

ثبتات‌های حدّی و مبنا، فقط توسط سیستم عامل و با به کارگیری دستورالعمل‌های ممتاز خاص، بارگذاری می‌شوند. از آن جایی که دستورات ممتاز فقط در مود مانیتور اجرا می‌شوند، و چون فقط سیستم عامل در مود مانیتور اجرا می‌شود، تنها سیستم عامل، رجیسترهای حدّی و مبنا را لود می‌کند. برنامه‌های کاربر از تغییر دادن مقادیر این رجیسترها، منع می‌شوند.

سیستم عامل در حال اجرا در مود مانیتور، دسترسی بلاشرط به هم حافظه مانیتور و هم حافظه کاربران دارد. این توانایی به سیستم عامل اجازه می‌دهد برنامه‌های کاربران را در فضای حافظه کاربران بارگذاری نماید، در صورت خطا از آن برنامه‌ها، رونوشت گیرد، پارامترهای فراخوانی‌های سیستم^۲ را تغییر دهد و بدانها دسترسی داشته باشد و غیره.

2.5.4. CPU Protection

۴.۵.۲. حفاظت CPU

قطعه سوم از معماهی حفاظت، این است که تضمین شود سیستم عامل کنترل را نگه می‌دارد. بایستی از یک لوب بی‌نهایت در برنامه کاربر، که منجر به عدم بازگشت کنترل به سیستم عامل می‌شود، جلوگیری شود. جهت انجام این هدف می‌توانیم از یک تایمر استفاده کنیم. تایمر می‌تواند طوری تنظیم شود که در پریود زمانی معینی، وقفه صادر نماید. پریود زمانی می‌تواند ثابت یا متغیر باشد. تایمر متغیر معمولاً با یک ساعت فرکانس ثابت و یک شمارنده، پیاده سازی می‌شود. سیستم عامل شمارنده را بست می‌کند. هر بار که ساعت تیک کند، شمارنده یک واحد کاهش می‌یابد. وقتی به صفر برسد، وقفه رخ می‌دهد. به عنوان مثال یک شمارنده ۱۰-بیتی با یک ساعت ۱ میلی ثانیه‌ای، اینتراتپ‌هایی از ۱ میلی ثانیه تا ۱۰۲۴ میلی ثانیه در فواصل ۱ میلی ثانیه‌ای صادر خواهد نمود.

قبل از برگرداندن کنترل به کاربر، سیستم عامل اطمینان می‌دهد که تایمر بست شده است. وقتی تایمر وقفه صادر کند، کنترل به طور خودکار به سیستم عامل منتقل می‌شود. دستورالعمل‌هایی که تایمر را تغییر دهند، بدیهی است که دستورالعمل‌های ممتاز محسوب می‌شوند.

بنابراین، می‌توانیم از تایمر استفاده کرده و از اجرای یک برنامه کاربر، به مدت طولانی پیشگیری نمائیم. یک تکنیک ساده آن است که مقدار اولیه شمارنده، همان میزان زمانی مجاز برنامه باشد. برای برنامه‌ای با ۷ دقیقه محدودیت زمان، مقدار اولیه شمارنده می‌تواند ۴۲۰ باشد و ساعت هر ثانیه بشمارد. وقتی شمارنده به صفر برسد، سیستم عامل برنامه را متوقف خواهد ساخت.

یک مورد استفاده معمول تر از تایمر، در مورد سیستم‌های اشتراک-زمانی است. مقدار اولیه تایمر N میلی ثانیه و N مقدار کوانتم زمانی هر پردازش می‌باشد. در پایان زمان، وقفه، سیستم عامل را فرا می‌خواند و سیستم عامل وظایف نظارت از قیل افزودن N به رکورد ویژه پردازش جهت تعیین میزان کل زمان اجرا شده (برای مقاصد حسابداری)، را انجام می‌دهد. سیستم عامل سایر تغییرات در رجیسترها و بافرها را برای پردازش بعدی آماده می‌نماید. (این روال به معنای برگردان ۱ متن نامیده می‌شود و در فصل ۴ غور می‌گردد). بعد از سوئیچ متن، برنامه بعدی از محلی که باقی مانده استفاده دیگر از تایمر، محاسبه زمان کنونی است. وقفه تایمر گذر پریودی را سیگنال می‌شود.

عامل اجازه می‌دهد برحسب زمان اولیه‌ای، به محاسبه ساعت کنونی پردازد. اگر وقفه‌ها هر یک ثانیه رخ دهند و ۱۴۲۷ وقفه از زمان پایه ۰:۰:۰۰ بعد از ظهر دریافت نماییم، در این صورت زمان کنونی ۱:۲۳:۴۷ بعد از ظهر خواهد بود. بعضی از کامپیوترها، زمان حقیقی را بدین ترتیب معین می‌کنند؛ البته محاسبات باید به دقت انجام گیرد تا وقت دقیق حاصل شود. زیرا زمان پردازش وقفه (و سایر زمان‌ها برای منع کردن وقفه‌ها) باعث می‌گردد که نرم‌افزار ساعت،

کندر شود. اکثر کامپیوترها، سخت افزار جداگانه ساعت زمان - روز^۱، مستقل از سیستم عامل، دارند.

2.6. General System Architecture

۲.۶.۲. معماری کلی سیستم

میل به بھبود بھرہوری سیستم عامل، منجر به توسعہ چند برنامگی و چند وظیفہ گی گردید. یعنی منابع سیستم مابین برنامہ ها و پراسسیس ها تقسیم شدند. اشتراک، مستقیماً در معماری پایه ای کامپیوتر، اثر گذاشت، و اجازه داد که سیستم عامل کنترل سیستم کامپیوتر و به ویژه O/I را به دست گیرد. اگر مایل به تحقق عمل پیوسته، سازگار و صحیح هستیم، کنترل باستی صورت گیرد.

جهت داشتن کنترل، توسعه دهنده گان اجرای مود - دوگانه را معرفی نمودند. این طرح مفهوم دستورالعمل های ممتاز را حمایت می کند. دستورالعمل های O/I و دستورات تغییر دهنده رجیستر های مدیریت - حافظه یا تایمر، همگی دستورالعمل های ممتاز در مورد مانیتور، هستند.

همان گونه که می توانید تصور کنید، دستورالعمل های متعدد دیگری، نیز به عنوان ممتاز طبقه بندی می شوند. به عنوان مثال، دستورالعمل halt، ممتاز است؛ یک برنامه کاربر نباید کامپیوتر را متوقف سازد. دستورالعمل های مجاز یا غیر مجاز نمودن وقفه ها هم، ممتاز هستند، به این دلیل که عمل درست تایمر و O/I، بستگی به توانایی پاسخگویی صحیح به وقفه، دارند. دستور سوئیچ از مود کاربر به مود مانیتور، ممتاز است و هر دستور تغییر بیت مود، نیز ممتاز محض می شود.

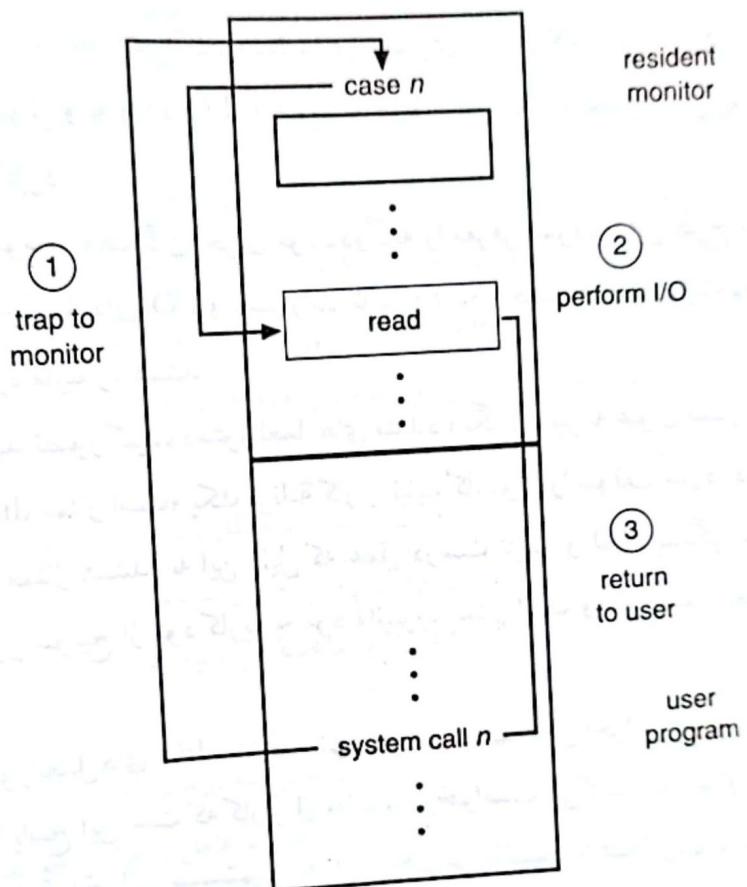
از آن جایی که دستورالعمل های O/I ممتازند، تنها توسط سیستم عامل اجرا می شوند. پس برنامه کاربر چگونه عمل O/I را انجام می دهد؟ پاسخ این است که کاربر از مانیتور درخواست می کند تا به نام او عمل O/I را انجام دهد. چنین درخواستی به نام فراخوانی سیستم (یا به نام فراخوانی مانیتور یا صدا زدن تابع سیستم عامل) شناخته شده است. احضار فراخوانی سیستم به شیوه های گوناگون، بر حسب قابلیت عمل پردازشگر مورد نظر، انجام می گیرد. معمولاً، فراخوانی سیستم به شکل یک تله سیستم عامل در محل ویژه ای در بردار وقفه، صورت می گیرد. این تله می تواند توسط دستور اصلی trap اجرا گردد، اگرچه بسیاری از سیستم ها (از جمله خانواده MIPS R2000) دستورالعمل خاص syscall را به کار می برنند.

وقتی یک فراخوانی سیستم صورت می گیرد، از دید سخت افزار همانند وقفه نرم افزاری تعبیر می شود. کنترل از طریق بردار وقفه به روایی سرویس در سیستم عامل، می رسد و بیت مود به مود مانیتور بست می شود. روای سرویس دهنده فراخوانی سیستم، بخشی از سیستم عامل است: مانیتور، دستورالعمل صادر کننده وقفه را تست می نماید؛ پارامتری نوع سرویس درخواستی را معین می کند. اطلاعات اضافی موردنیاز، می تواند در رجیستر ها، یا پشته، یا در

1. time-of-day clock

۵۰ ♦ فصل ۲: ساختمان‌های سیستم کامپیوتر

حافظه (که اشاره گر به مکان حافظه، در رجیسترها ی عبور داده می‌شود) منتقل شود. مانیتور بررسی می‌کند تا پارامترها درست و قانونی باشند. درخواست را اجرا می‌کند، و کنترل را به دستور بعد از فراخوانی سیستم، برمی‌گرداند (شکل ۹-۲).



شکل ۹-۲: استفاده از فراخوانی سیستم برای اجرای I/O

■ تمرینات

۱.۰ پیش واکشی^۱، روش روی هم انداختن O/I یک کار و محاسبات خود کار، می باشد. ایده، سهل است. بعد از اتمام عمل خواندن، زمانی که کار عمل بر روی داده را آغاز می کند، وسیله ورودی راهبری می شود تا فوراً شروع به خواندن بعدی نماید. پس هم CPU و هم دستگاه O/I مشغولند. با شناس، اگر، کار برای قلم داده بعدی آماده باشد، دستگاه ورودی هم خواندن آن قلم داده را تمام کرده است. CPU، سپس، پردازش را شروع می کند در حالی که دستگاه ورودی داده دیگر بعدی را می خواند. ایده همانندی در مورد خروجی می تواند به کار رود. در این حالت، کار، داده ای را تولید می کند که در بافری نگهدارشته می شود تا دستگاه خروجی بتواند آن را قبول کند.

طرح پیش واکشی و طرح اسپولینگ را با یکدیگر مقایسه نمایید.

۲.۰ چگونه تفاوت مودهای کاربر و مانیتور به عنوان یک محک ابتدایی حفاظت (امنیت) سیستم عمل می کند؟

۳.۰ تفاوت های ^{۱۰} و وقفه چیستند؟ کاربرد هر یک چیست؟

۴.۰ DMA برای چه نوع عملی مفید است؟ توضیح دهید.

۵.۰ کدام یک از دستورالعمل های زیر ممتاز است؟

d. قدغن کردن وقوع وقفه

a. مقدار تایمر را سیستم کردن

e. برگشت از حالت کاربر به حالت مانیتور

b. خواندن ساعت

c. پاک کردن حافظه

۶.۰ بعضی از سیستم های کامپیوتری، وجه ممتاز عمل را در سخت افزار، ارائه نمی دهند. آیا می توان سیستم عامل دقیقی برای چنین سیستم های طراحی نمود؟ بحث کنید.

۷.۰ بعضی سیستم های اولیه، سیستم عامل را بدین گونه حفاظت می کردند که، کد آن را در بخشی از حافظه که نه توسط کاربر و نه توسط خود سیستم عامل نمی توانست تغییر کند، قرار می دادند. دو شکل منشعب از این طرح، را تشریح کنید.

۸.۰ حفاظت سیستم عامل در اطمینان از عملکرد درست سیستم کامپیوتری، اساسی است. این حفاظت، دلیل محدودیت های کمینه ای را نیز به کاربر، اعمال نماییم.

محفوظت زیر، اعمالی است که به طور طبیعی محافظت شده اند. مجموعه کمینه دستورالعمل هایی که باید محفوظت شوند، چیست؟

- e. از حافظه مانیتور دستورالعملی را واکشی کنید.
 - f. تایمر وقفه را آغاز کنید.
 - g. تایمر وقفه را متوقف کنید.
 - a. به مود کاربر تغییر دهید.
 - b. به مود مانیتور تغییر دهید.
 - c. از حافظه مانیتور بخوانید.
 - d. در حافظه مانیتور بنویسید.
- ۹.۲. چه موقع حافظه‌های کش مفیدند؟ چه مشکلاتی را حل می‌کنند؟ چه می‌آورند؟ اگر کش بتواند به اندازه وسیله‌ای که آن را نهانی می‌کند (مثلاً، کشی به اندازه خود دیسک) باشد، چرا آن را نسازیم و وسیله را حذف ننماییم؟

۱۰.۲. نوشتن سیستم عاملی که بتواند علی‌رغم برنامه کاربر بدخواه یا خطایابی نشده، کار کند، نیازمند کمک سخت‌افزار است. سه کمک سخت‌افزاری در نوشتن سیستم عامل را نام برد، و شرح دهید که چگونه آن را حفاظت می‌کنند.

۱۰.۳. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۴. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۵. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۶. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۷. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۸. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۹. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۱۰. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۱۱. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟

۱۰.۱۲. در نوشتن سیستم عامل کدام این دو مسئله را برای نظریه کنید؟