بسمه تعالی

جزوه درس سیستم عامل (2)

ویژه دانشجویان دوره کاردانی پیوسته و ناپیوسته
دانشگاه آزاد اسلامی واحد برازجان

نام استاد : مهندس موسوی

تعریف سیستم عامل

مجموعه ای از برنامه ها را که موجب راه اندازی و استفاده از کامپیوتر می شوند " سیستم عامل Operation System (OS) گویند.

سیستم عامل بدون شک مهمترین نرم افزار در کامپیوتر است . پس از روشن کردن کامپیوتر اولین نرم افزاری که مشاهده می گردد سیستم عامل بوده و آخرین نرم افزاری که قبل از خاموش کردن کامپیوتر مشاهده خواهد شد، نیز سیستم عامل است . سیستم عامل نرم افزاری است که امکان اجرای تمامی برنامه های کامپیوتری را فراهم می آورد. سیستم عامل با سازماندهی ، مدیریت و کنترل منابع سخت افزاری امکان استفاده بهینه و هدفمند آنها را فراهم می آورد . سیستم عامل فلسفه بودن سخت افزار را بدرستی تفسیر و در این راستا امکانات متعدد و ضروری جهت حیات سایر برنامه های کامپیوتری را فراهم می آورد.

وظایف سیستم عامل:

سیستم عامل دو وظیفه اصلی دارد:

1- وظیفه اول سیستم عامل مدیریت منابع (Resource Management)می‌باشد، یعنی سیستم عامل باعث استفاده بهینه و سودمند (اقتصادی) از منابع سیستم می‌گردد . منظور از منابع پردازنده‌ها ,حافظه‌ها ،دیسک ها موس ها ، چاپگرها ، فایلها ، پورت ها و غیره هستند. یک سیستم کامپیوتری منابع نرم افزاری و سخت افزاری بسیار دارد که ممکن است در حین اجراء برنامه لازم باشند ,سیستم عامل همانند مدیر منابع عمل کرده و آنها را بر حسب نیاز به برنامه‌های مشخصی تخصیص می‌دهد.

2- وظیفه دوم سیستم عامل ساده کردن کار با کامپیوتر است . این بدان معناست که مثلاً کاربر یا برنامه نویس بدون درگیر شدن با مسائل سخت افزاری دیسک ها به راحتی فایلی را بر روی دیسک ذخیره و حذف کند. در صورت عدم وجود سیستم عامل کاربرو یا برنامه نویس می‌بایست آشنایی کاملی با سخت افزارهای مختلف کامپیوتر (مثل مانیتور ،فلاپی ،کی برد و غیره) داشته باشدو روتین‌هایی برای خواندن و یا نوشتن آنها به زبانهای سطح پائین بنویسد.سیستم عامل یک ماشین توسعه یافته (Extended machine) که واقعیت سخت افزار را از دید برنامه نویسان مخفی می‌سازد.

انواع سیستم های عامل

سیستمهای عامل انواع گوناگون دارند که با توجه به اندازه کامپیوتر و نوع کاربرد های آن برخی از آنها بسیار ساده و برخی دیگر پیچیده است .

1. **سیستم عامل تک برنامه ای single program :**

برخی از کامپیوتر ها میتوانند در یک لحظه فقط به پردازش یک برنامه بپردازند . سیستم عامل های این نوع کامپیوترها می توانند برنامه را بارگذاری و اجرا کنند و یا اطلاعات را به دستگاه جانبی بفرستند یا از آن دریافت کنند و دستورات مخصوص خود را به اجرا درآورند .

2. سیستم عامل های چند برنامه ای multi programming :

برای جلوگیری از تلف شدن وقت واحد پردازشگر این سیستم ها طوری طراحی شده اند که می توانند اجرای چند برنامه را به طور همزمان بر عهده بگیرند . انجام این کار بدبن صورت است که در هر لحظه چند برنامه در داخل حافظه موجود هستند هر کدام از این برنامه ها در مرحله اجرای خاص خود قرار دارند .این برنامه ها به صورت قسمت قسمت مطابق با احتیاجات ورودی و خروجی خود پردازش می شوند .

می توان برای پردازش برنامه ها اولویت هائی را نیز در نظز گرفت در این حالت برنامه هایی که دارای اولویت برای اجرا هستند زود تر از دیگر برنامه ها پردازش خواهند شد

تاریخچه سیستم عامل :

جهت بررسی تاریخچه سیستم عامل باید تاریخچه معماری کامپیوتر ها را در نظر گرفت که سیستم عامل ها بر روی آنها قابل اجرا بودند :

نسل اول 1955-1945 – استفاده از لامپهای خلا در ساخت کامپیوتر – سیستم عاملی وجود نداشت

نسل دوم 1965-1955 – استفاده از ترانزیستورها – ایده ساخت برنامه های کوچک سیستمی

نسل سوم 1975-1965 -- ساخت مدارات مجتمع IC - تولید سیستم عاملهای اشتراک زمانی

نسل چهارم 1955-1975 -- افزایش سرعت سخت افزاری و نرم افزاری -- یستم عامل های چند پردازنده ای

تقسیم بندی سیستم عامل از جهت ارتباط با دستگاهای ورودی / خروجی

1) سیستم های روی خطی : (On - Line) : سیستم هایی که در آنها پردازنده مستقیماً با دستگاهای I/O در ارتباط است . به دلیل کند بودن I/O بیکاری پردازنده (Cpu) زیاد بوده و بهره وری سیتم کاهش می یابد .

2) سیستم های off-line : سیستم هایی که در آنها اطلاعات از طریق یک حافظه جانبی سریعتر نسبت به I/O ، به پردازنده داده شده و خروجی پردازنده هم از طریق همان نوع حافظه جانبی مثلاً TAPE به واحد خروجی ارسال می گردد .

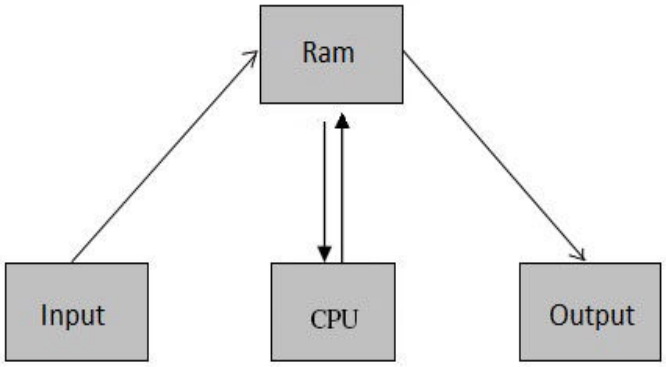
تقسیم بندی سیستم عامل از جهت ارتباط با کاربر :

1- سیستم های دسته ای (Batch Systems) : در این سیستم ها پردازش در دو مرحله انجام می شود . ابتدا یک دسته **L** تایی از کارهایی که نیاز مشابه دارند دریافت می شوند . در مرحله دوم آن نیاز یا منبع در اختیار کارها قر ار گرفته و آنها پشت سرهم اجرا می شوند . پردازش دسته ای در نسل دوم سیستم های عامل مطرح شدند . در سیستم های دسته ای ، استفاده اشتراکی از منابع نداریم .

2- محاوره ای (Interactive) : کاربر بصورت مستقیم با کامپیوتر در ارتباط است . دستوراتی را وارد می کند و منتظر پاسخ می ماند .

سیستم بافرینگ (Buffering) :

ناحیه ای از حافظه است که جهت ایجاد هماهنگی بین وسایل ورودی/خروجی (که دارای سرعت پایین هستند) و پردازنده استفاده میشود. بافرینگ امکان همزمانی پردازش و ورودی/خروجی یک کار را به کمک حافظه اصلی فراهم میکند.



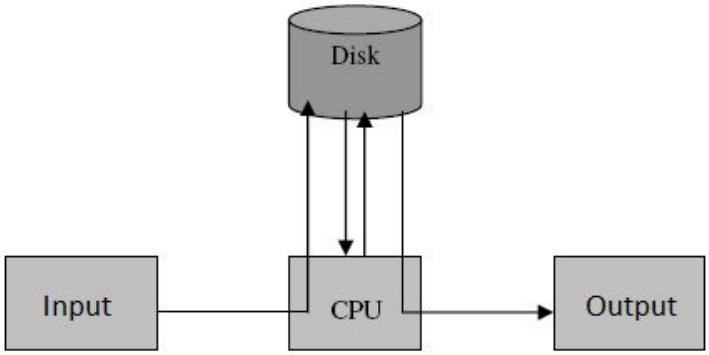
سیستم اسپولینگ (Spooling) :

1- Offline Spooling

در این روش ابتدا کلیه اطلاعات بوسیله دستگاه کارتخوان یک کامپیوتر کوچک بر روی نوار (Type) ذخیره میشد و سپس اپراتور نوار را برداشته و روی کامپیوتر اصلی که محاسبات را انجام میداد نصب میکرد و اطلاعات توسط کامپیوتر اصلی از نوار خوانده میشد سپس اطلاعات خروجی به روی نوار دیگر ذخیر میشد که اپراتور مجبور بود نوار خروجی را برداشته و برای چاپ به روی کامپیوتر اول منتقل نماید به این روش Offline Spooling گفته میشود.

2- Online Spooling

ویژگیهاي سیستم عامل نسل سوم Online Spooling است که معمولا همراه چند برنامه گي استفاده مي شود. در این روش داده ها از ورودی کاراکتر به کاراکتر در بافري در حافظه قرار گرفته و سپس به صورت بلوکي بر روي دیسک نوشته مي شود. سپس اطلاعات توسط CPU بصورت بلوکی از دیسک خوانده شده و به خروجی ارسال میگردد. سیستم اسپولینگ امکان هم زمانی پردازش و ورودی/خروجی چند کار را به کمک حافظه جانبی(دیسک سخت) انجام میدهد



فرق بافرینگ و اسپولینگ چیست؟

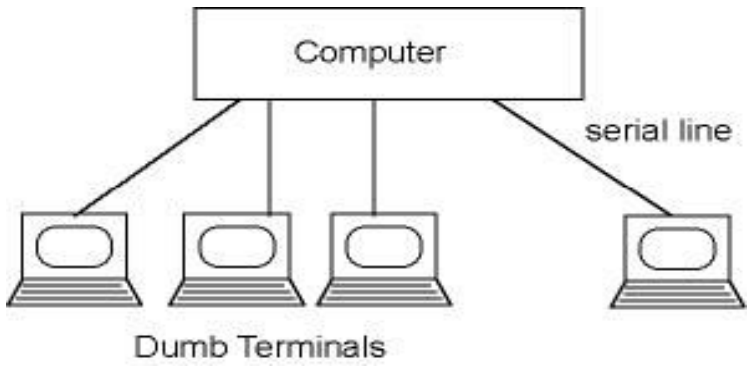
بافرینگ امکان همزمانی پردازش و ورودی/خروجی یک کار را به کمک حافظه اصلی فراهم میکند. سیستم اسپولینگ امکان هم زمانی پردازش و ورودی/خروجی چند کار را به کمک حافظه جانبی(دیسک سخت) انجام میدهد

سیستم اشتراک زمانی (Time Sharing):

این سیستم‌ها از اوایل سالهای 1970 در نسل سوم کامپیوترها معمول شدند. سیستم اشتراک زمانی در واقع تعمیم سیستم چند برنامه ای است .

در سیستم‌های چند برنامگی کاربر ارتباطی با کامپیوتر نداشت و خطایابی برنامه‌ها مشکل بود چرا که زمان برگشت نسبتاً طولانی اجازه آزمایش کردنهای متعدد را نمی‌داد. در سیستم اشتراک زمانی کاربر به کمک دو ترمینال (Terminal)که شامل کی برد (برای ورودی) و مونیتور (برای خروجی) است با کامپیوتر به صورت محاوره‌ای (interactive) رابطه بر قرار می‌سازد .کاربر مستقیماً دستوراتی را وارد کرده و پاسخ سریع آن را روی مونیتور دریافت می‌کند. در این سیستم‌ها چندین کاربر به کمک ترمینالهایی که به کامپیوتر وصل است همزمان می‌توانند از آن استفاده کنند . در سیستم اشتراک زمانی فقط یک پردازنده وجود دارد که توسط مکانیزمهای زمانبندی بین برنامه‌های مختلف کاربرها با سرعت زیاد (مثلاً در حد میلی ثانیه) سوئیچ می‌شودو بنابراین هر کاربر تصور می‌کند کل کامپیوتر در اختیار اوست .در اینجا تأکید بر روی میزان عملکرد کاربر است یعنی هدف فراهم کردن وسایل مناسب برای تولید ساده نرم افزار و راحتی کاربرد می‌باشد و نه بالا بردن میزان کاربرد منابع ماشین . کاربر می‌تواند در هر زمان دلخواه برنامه خود را آغاز یا متوقف سازد و یا برنامه را به صورت قدم به قدم اجراء و اشکال زدایی (debug) کند . سیستم‌های دسته‌ای برای اجرای برنامه‌های بزرگ که نیاز محاوره‌ای کمی دارند مناسب است ولی سیستم‌های اشتراک زمانی برای مواردی که زمان پاسخ کوتاه لازم است , استفاده می‌شوند .

در زمانی که کاربری در حال تایپ برنامه‌اش یا فکر کردن روی خطاهای برنامه اش می‌باشد CPUبه برنامه کاربر دیگری اختصاص یافته تا آن را اجراء کند.



تعریف پردازش (Process) :

مهمترین مفهوم در هر سیستم عامل فرآیند یا پردازش (process) است . تمامی نرم افزارهای کامپیوتر از جمله سیستم عامل به تعدادی از پردازشها سازماندهی و تقسیم بندی می‌شوند.

پردازش یک برنامه در حال اجراست که منابعی از سیستم به آن تخصیص داده شده است (شامل رجیستر ، حافظه ، فایلها و دستگاهها). که دارای ساختار خاصی بنام (Process Control Block) PCB میباشد. کلیه اطلاعات مربوط به هر پردازش ، در یکی از جداول سیستم عامل به نام جدول PCB ذخیره میشود..

مثلاً در یک کامپیوتر کاربران متعددی ممکن است در حال اجرای نسخه‌های متعددی از برنامه ویرایشگر باشند یا مثلاً یک کاربر می‌تواند چند نسخه از برنامه ویرایشگر را همزمان اجراء کند ، در این حال هر کدام از آنها یک پردازش جداگانه‌اند و اگر چه بخش متن شان (کدشان)یکسان است ولی بخش داده‌هایشان متفاوت می‌باشد.

در سیستمها روشی مورد نیاز است تا در حین کار بتوان پروسس‌هایی را ایجاد کرد یا از بین برد لذا در سیستم عامل ، پروسس‌ها توسط یک فراخوان سیستمی پدید می‌آیند، این فراخوانی یک پردازش فرزند تولید می‌کند که نسخه‌ای دقیقاً یکسان با پروسس پدر خواهد بود. به همین ترتیب پردازش فرزند نیز می‌تواند فراخوان سیستمی را برای یک پردازش جدید اجراء کرده و لذا سیستم می‌تواند درختی از پروسس‌ها داشته باشد. بدیهی است هر پروسس فقط یک پدر دارد ولی می‌تواند صفر یا چندین فرزند داشته باشد.

اطلاعات موجود در PCB عبارتند از :

حالت جاری پردازش
شماره شناسایی پردازش
اولیت پردازش
نشانی حافظه پردازش
نشانی محل برنامه پردازش بر روی دیسک
نشانی سایر منابع پردازش
محلی برای حفظ ثباتها

سیستم های چند پردازندهای Multi Processing

کامپیوترها میتوانند بجای یک cpu ، چندین cpu داشته باشند که در اینصورت به آنها سیستم multiprocessing میگویند . جهت استفاده از ا ین سیستمهای نیاز به یک سیستم عامل خاص میباشد . که بتواند چندین برنامه را به صورت موازی واقعی روی آنها اجراء کند.

در سیستم های چند پردازنده ای ، cpuها باید بتوانند از حافظه ، امکانات ورودی و خروجی و گذرگاه سیستم بصورت اشتراکی استفاده کنند . **مزایای استفاده از این سیستمها عبارتند از :**

1- زیاد شدن توان عملیاتی Throughput : یعنی تعداد کارهایی است که در یک بازه زمانی تمام میشوند
2- صرفه جویی در هزینه ها
3- تحمل پذیری در برابر خطا : دارای امنیت بیشتری است . خراب شدن یک پردازنده سبب توقف سیستم نمیشود

یک پردازش میتواند در چند حالت اصلی قرار گیرد :

1- حالت آماده (Redy State) : حالتی که پردازش همه منابع سیستم را در اختیار دارد و فقط منتظر CPU میباشد
2- حالت منتظر(Waiting) : حالتی که در آن پردازش منتظر بدست آوردن یک منبع از سیستم میباشد.
3- حالت اجرا (Executing) : همه منابع و cpu را در اختیار دارد و در حال اجراست
4- حالت معلق (Suspending) اگر پردازش به مدت زیادی در حالت آماده ، منتظر CPU باشدو CPU به آن توجهی نکند در آن صورت پردازش موقتا بر روی دیسک انتقال میابد.

صف های سیستم :

- صف کار (Job Queue) : در این صف برنامه هایی که منتظر تبدیل شدن به پردازش هستند قرار میگیرند. مدیریت این صف به عهده زمانبند کار است
- صف آماده (Redy Queue) : در این صف پردازش هایی که منتظر cpu هستند قرار میگیرندو توسط زمانبند کوتاه مدت مدیریت میشوند
- صف انتظار (Waiting Queue) : در این صف پردازش هایی که منتظر وسایل جانبی I/O هستند قرار میگیرند.

انواع زمانبندی:

1- زمانبندی کوتاه مدت :

پردازش در حال اجرا باید از پردازنده و از صف پردازش های آماده اجرا خارج شود و بر اساس الگوریتم زمانبندی ، یک پردازش دیگر انتخاب شده و بار (Load)شود .این مرحله تعویش پردازش توسط مدیر زمانبند کوتاه مدت انجام میشود. این زمانبندی با توجه به کوتاه بودن برش زمانی ، در فواصل کم با فرکانس بالا انجام میشود.

2- زمانبندی میان مدت :

در بعضی شرایط به دلیل زیاد شدن پردازش های موجود در چرخه حالت پردازش ها (آماده – اجرا – منتظر) و در نتیجه کم شدن حافظه آزاد سیستم و کاهش کارایی ، بهتر است تعدادی از پردازش ها را از

حالت فعال خارج کرده و پس از اتمام اجرای تعدادی از پردازش ها ، اجرای آنها ادامه یابد. عمل انتقال پردازش های آماده موجود در حافظه اصلی ، به حافظه جانبی به منظور کاهش بار سیستم Swap out و انتقال دوباره پردازش های آماده از حافظه جانبی به حافظه اصلی و فعال شدن دوباره آنها Swap in نام دارد که هر دو توسط بخشی از سیستم عامل به نام مدیریت زمانبند میان مدت انجام میشود.

3- زمانبند بلند مدت :

انتخاب یکی از کارهای موجود در سیستم جهت تبدیل شدن به پردازش . چون در فواصل طولانی انجام میگیرد زمانبندی بلند مدت گفته میشود.

عموما کارها در دو دسته قرار می گیرند:

1 – کارهای I/O bound (I/O Limited) : کارهائی که بخش زیادی از اجرای آنها در ارتباط با دستگاههای ورودی / خروجی بوده و محاسبات زیادی ندارد . مثل برنامه ای که می بایست کارنامه ی دانشجویان را چاپ کند .

2 – کارهای CPU bound (CPU Limited) : کارهائی که حجم زیادی محاسبات و بخش عمده نیاز آنها برای اجرا وقت پردازنده است . مثل برنامه ای که می بایست یک دستگاه معادله ی چتر مجهول را حل کند .
زمانبند بلند مدت می بایست تلفیق مناسبی از کارهای I/O dound , CPU bound را انتخاب کند .

حالت پایدار : زمانی که نرخ ایجاد پردازش ها برابر با نرخ خروج یا خاتمه پردازش ها باشد . بنابراین زمانبند بلند مدت سعی می کند که سیستم به حالت پایدار برسد .

الگوریتم های زمانبند کوتاه مدت (پردازنده) :

عمل زمانبندی کوتاه مدت براساس الگوریتم های زمانبندی انجام می شود . این الگوریتم ها سعی در بر آورده کردن معیارهای زیر را دارند :

1. عدالت : یعنی سهم پردازش ها از CPU به طور منصفانه باشد .
2. افزایش کارئی : هدف از کارائی مشغول بودن CPU به طور ایده آل است .
3. افزایش گذردهی : منظور تعداد پردازش های انجام شده در واحد معینی از زمان می باشد .
4. کاهش زمان پاسخ : زمان ما بین مطرح شدن یک پردازش تا اولین اجرای آن پردازش می باشد .

این اهداف کاملاً با هم در تناقض اند بنابراین می بایست سعی شود مصالحه ای بین این اهداف صورت گیرد .

زمان انتظار (waiting Time) : مدت زمانی را که پردازش در حالت آماده است و منتظر CPU می باشد را زمان انتظار گویند .

زمان اجرا (Runng Time) : مدت زمانی را که پردازش بر روی CPU در حال اجراست را زمان اجرا گویند .

زمان I/O : مدت زمانی است که پردازش نیاز به عمل I/O دارد .

زمان گردش کار (Turnatound Time) : مدت زمان ما بین ورود یک پردازش به سیستم تا اجرای کامل یک پردازش را زمان گردش کار گویند .

الگوریتم های زمانبندی به دو دسته تقسیم می شوند:

1. **انحصاری یا انقطاع نا پذیر (Preemptive)** : در این الگوریتم ها تا زمان خاتمه پردازش و یا زمان نیاز به عمل I/o نمی توان CPU را از پردازش در حال اجرا گرفت و به پردازش دیگری انتساب داد .

2. **غیر انحصاری یا انقطاع پذیر (Non Preemptive)** : در این الگوریتم ها درپایان برش زمانی (Time slice) و یا تغییر شرایط سیستم مدیر زمانبندی می تواند کنترل پردازنده را از یک پردازش در حال اجرا گرفته و به پردازش دیگری بدهد .

الگوریتم الویت با اولین ورودی – FCFS – (Frist In Frist Out - Frist Come Feist Service)

این الگوریتم به سادگی هر پردازش ورودی را در صف FIFO قرار داده و از سر صف اجرایش را شروع می کند و اجرای پردازش ها را به طور انحصاری انجام می دهد .

مزایا :

سادگی اجرا – عملی بودن

معایب :

1 – زیاد بودن میانگین زمان انتظار (زمان گردش کار)

2 – اقطاع ناپذیری و غیر قابل استفاده بودن در سیستم های اشتراک زمانی است .

الگوریتم الویت با کوتاهترین کار – SJF (Shortest Job Frist) :

این الگوریتم از بین پردازش های موجود در صف آماده ، پردازش ای را جهت اجرا انتخاب می کند که زمان اجرای کمتری نیاز داشته باشد و پردازش در حال اجرا اجرایش را به صورت انقطاع ناپذیر ادامه می دهد .

معایب :

1 – نیاز به داشتن اطلاعات مقاطع زمانی مورد نیاز قبل از شروع اجرا

2 – احتمال به تعویق افتادن کارهای طولانی

3 – انقطاع ناپذیری

مزایا : دارا بودن کمترین زمان برگشت یا زمان انتظار مابین تمام الگوریتمها

الگوریتم الویت با کمترین زمان باقیمانده – SRT (Shortest Remaining Time)

در این الگوریتم انتخاب براساس کمترین زمان مورد نیاز برای کامل شدن صورت میگیرد . این الگوریتم غیر انحصاری بوده و با ورود هر پردازش به صف آماده ، بررسی زمان باقیمانده پردازش ها انجام می گیرد ، اگر پردازش تازه وارد شده ، زمان کمتری برای کامل شدن لازم دارد ، پردازنده در اختیار آن قرار می گیرد .

معایب : احتمال تعویق کارهای طولانی – نیاز به دانستن زمان مورد نیاز پردازش ها

مزایا : انقطاع پذیری – زمان پاسخ نسبتاً مناسب

مثال : در سیستمی سه پردازش P_2, P_1, P_0 مطابق جدول زیر با زمان های ورود و اجرای داده شده قرار دارند مطلوب است محاسبه میانگین زمان انتظار و میانگین زمان گردش کار در حالت های زیر

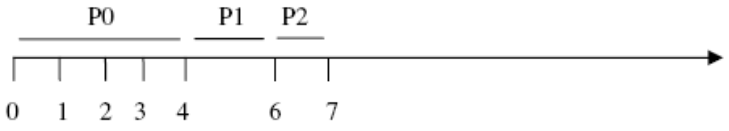
پردازه	زمان رسیدن	زمان اجرا
P_0	t	4
P_1	t+1	2
P_2	t+2	1

الف : برای زمانبندی پردازش ها از الگوریتم FCFS استفاده شود .

ب : برای زمانبندی پردازش ها از الگوریتم SJF استفاده شود .

ج : برای زمانبندی پردازش ها از الگوریتم SRT استفاده شود .

الف)



$$\text{میانگین زمان انتظار} = \frac{\text{مجموع زمان انتظار تمام پردازه ها}}{\text{تعداد پردازه ها}} \qquad \text{میانگین زمان گردش کار} = \frac{\text{زمان اجرای پردازه+زمان انتظار}}{\text{تعداد پردازه ها}}$$

$$\text{میانگین زمان گردش کار} = \frac{(0+4)+(3+2)+(4+1)}{3} = \frac{14}{3} \qquad \text{میانگین زمان انتظار} = \frac{0+3+4}{3} = \frac{7}{3}$$

ب) در این شرایط پردازش ای شروع به اجرا شود تا انتها ادامه می یابد در مرحله ی بعد پردازش ای برای اجرا انتخاب می شود که کمترین زمان اجرا را دارد بنابراین اول P0 (در این لحظه فقط همین را داریم) تا لحظه ی 4 اجرا می شود در این لحظه دو تا پردازش آماده داریم که P2 زمان کمتری را برای اجرا می خواهد پس تا لحظه 5 اجرا می شود از 5 تا 7 زمان اجرای p1 می باشد .

زمان انتظار : P0=0s , P1=4s , P2=2s

$$\text{میانگین زمان انتظار} = \frac{0+4+2}{3} = 2 \qquad \text{میانگین زمان گردش کار} = \frac{(0+4)+(4+2)+(2+1)}{3} = \frac{13}{3}$$

ج) در این شرایط در تمامی لحظه های اجرا اگر پردازشی از راه برسد که زمان اجرایش کمتر از زمان باقیمانده پردازش در حال اجرا باشد پردازنده را در اختیار گرفته و شروع به اجرا می کند .

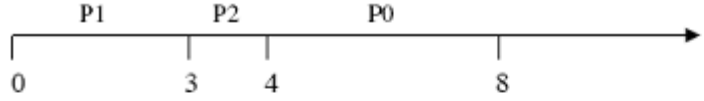
زمان انتظار : P0=3s ,p1=0s,p2=1S

$$\text{میانگین زمان گردش کار} = \frac{(3+4)+(0+2)+(1+1)}{3} = \frac{11}{3} \qquad \text{میانگین زمان انتظار} = \frac{3+0+1}{3} = \frac{4}{3}$$

مثال : مثال قبل را با جدول زیر حل کنید :

پردازه	زمان رسیدن	زمان اجرا
P_0	0	4
P_1	0	3
P_2	1	1

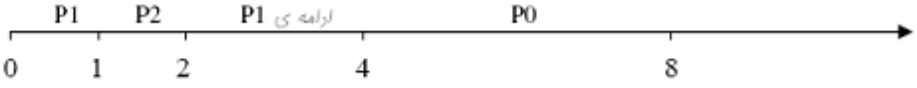
ب) SJF



میانگین زمان انتظار $AWT = \frac{4+0+2}{3} = 2$

میانگین زمان گردش $ATT = \frac{(4+4)+(0+3)+(2+1)}{3} = \frac{14}{3}$

ج) SRT



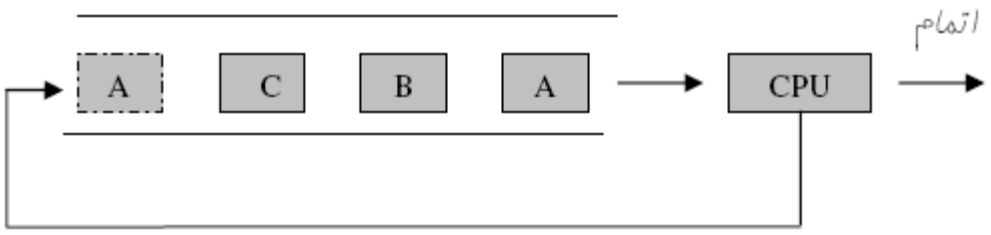
$AWT = \frac{4+1+0}{3} = \frac{5}{3}$

$ATT = \frac{(4+4)+(1+3)+(0+1)}{3} = \frac{13}{3}$

الگوریتم نوبتی (Round Robin : RR) :

در این الگوریتم با در نظر گرفتن یک برهه زمانی (time slice) یا کوانتوم زمانی پردازش های موجود در صف آماده هر کدام به اندازه این برهه زمانی CPU را بدست آورده به طوری که اگر در برهه زمانی شان اجرای خود را به اتمام نرسانند مجدداً به انتهای صف منتقل می شوند .

این الگوریتم بیشتر در روش های اشتراک زمانی (Time sharing) استفاده می شود .



انتظار برای I/O یا پایان یافتن برش زمانی

مزایا :

1 – زمان پاسخ نسبتاً مناسب (نوبه اجرا به همه می رسد)

2 – غیر انحصاری بودن

3 – عدم نیاز به دانستن مقاطع زمانی پردازش ها

معایب :

1 – نیاز به دقت در تنظیم برش زمانی

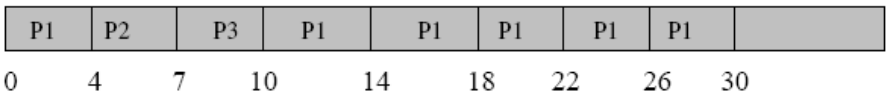
2 – کاهش کارائی پردازنده به خاطر زمان های تعویض متن در برهه های زمانی خیلی کوتاه

3 – شباهت به الگوریتم FCFS برای برش های زمانی طولانی

مثال : سه پردازش زیر را در نظر بگیرید ، اگر از کوانتوم زمانی 4 میلی ثانیه استفاده شود ، میانیگن زمان انتظار و میانیگین زمان گردش کار را بیابید . (هز سه در لحظه صفر وارد شده اند ولی اولیت به ترتیب با شماره است) .

پردازه	زمان اجرا
p_1	24
p_2	3
p_3	3

حل)نمودار پردازشها بصورت زیر خواهد بود:

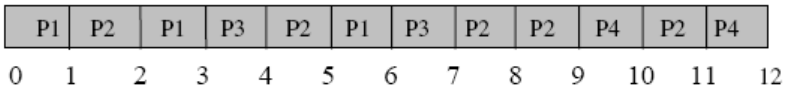


میانگین زمان گردش کار $= \frac{(6+24)+(4+3)+(7+3)}{3} = \frac{47}{3} ms$ میانگین زمان انتظار $= \frac{6+4+7}{3} = \frac{17}{3} ms$

مثال :

چهار پردازنده مطابق جدول زیر در سیستم وجود ندارد . اگر از روش RR با برش زمانی lms استفاده شود و از سر بار ناشی از تعویض متن فرایندها صرف نظر شود میانگین زمان انتظار پردازش ها چقدر است (فرض شود پردازش ای که از راه می رسد به ابتدای صف منتقل می شود) .

پردازش ها	زمان ورود	زمان پردازش
p_1	0	3
p_2	1	5
p_3	3	2
p_4	9	2



$$\text{متوسط زمان انتظار} = \frac{3+5+2+1}{4} = \frac{11}{4} ms$$

الگوریتم زمانبندی الویت دار

الویت با بالاترین نسبت پاسخ (Highest Respones Ratio Next – H RN)

$$\text{الویت} = \frac{\text{زمان انتظار} + \text{زمان سرویس}}{\text{زمان سرویس}}$$

الگوریتم های الویت دار با در نظر گرفتن یک الویت برای هر کدام از پردازش های موجود در سیستم ، پردازش ای که بالاترین الویت را داشته باشد اجرایش را به صورت انحصاری انجام می دهد ، الویت هائی که می توان در این الگوریتم در ظنر گرفت عبارتند از :

1 – نسبت مقاطع زمانی ورودی یا خروجی به مقاطع زمانی پردازنده (CPU Burst I/O Buest)

2 – محدودیت زمانی ، نیازهای حافظه

3 – تعداد فایل های باز شده

4 – اهمیت پردازش و بخش ارائه دهنده کار

5 – الویت نسبت پاسخ (فرمول بالائی)

مزایا :

1 – طبق این الگوریتم پردازش های کوتاه در ابتدا الویت بالا پیدا می کنند ، چون زمان سرویس (زمان مورد نیاز جهت اجرا) کمتری نیاز دارند .

2 – پردازش هائی که مدت زیادی منتظر مانده اند نیز الویت بالا پیدا می کنند ، پس این شانس را پیدا می کنند که اجرا شوند .

معایب :

1 – می بایست زمان سرویس را از قبل معلوم کرد .

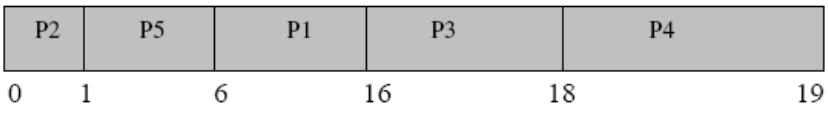
2 – انحصاری است .

مثال :

پردازش های زیر را در نظر بگیرید که همگی در لحظه ی صفر به ترتیب داده شده رسیده اند میانگین زمان انتظار را بیابید و جهت زمانبندی پردازش ها از الگوریتم الویت دار استفاده کنید و پردازش ای که عدد الویت آن کمتر است ، الویت بالاتری دارد .

حل) نمودار زمانی به شکل زیر خواهد بود

پردازه ها	زمان اجرا	الویت
p_1	10	3
p_2	1	1
p_3	2	3
p_4	1	4
p_5	5	2



$$\text{میانگین زمان انتظار} = \frac{6+0+16+18+1}{5} = \frac{41}{5} ms$$

الگوریتم صف چند سطحی (Multi Level Queue : MLQ) :

در الگوریتم صف چند سطحی پردازش ها در صف های مختلف ، که هر صف الویت خاصی دارد قرار می گیرند و در صف های مختلف الگوریتم های زمانبندی مختلفی استفاده می شود ، پارامترهائی که در این الگوریتم می بایست مشخص نمود عبارتند از :

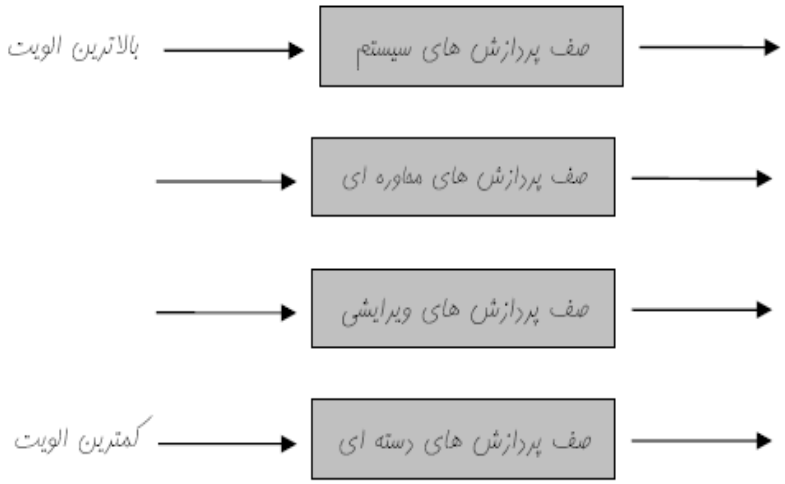
1 – تعداد صف ها

2 – الگوریتم زمانبندی استفاده شده در هر صف

3 – الویت صف ها نسبت به هم

اگر در صفی با الویت بالا پردازش ای وجود داشته باشد CPU در ابتدا پردازش های آن صف را سرویس داده و در صورتی که صف های الویت بالاتر خالی شود به سراغ صف الویت پایین می رود .

به عنوان مثال سیستمی می تواند 4 صف آماده با الویت های زیر داشته باشد .



الگوریتم صف باز خورد چند سطحی (Multi Level Feedback Queue: MFQ)

این الگوریتم مانند الگوریتم MLQ می باشد با این تفاوت که در این الگوریتم امکان حرکت پردازش ها بین صف های مختلف نیز وجود دارد ، در این الگوریتم علاوه بر مشخص نمودن پارامترهای MLQ موراد زیر نیز می بایست مشخص شود .

1 – چه موقع یک پردازش از صف بالا به صف پایین مهاجرت می کند .

2 – چه موقع یک پردازش از صف پایین به صف بالا مهاجرت می کند .

بن بست پردازش ها (Process Deadlocks)

اگر مجموعه ای از پردازش ها در سیستم وقوع عملی باشند که توسط دیگری انجام شود و هیچ کاری در سیستم پیش نرود ، گوئیم سیستم دچار بن بست شده است .

شرایط وقوع بن بست : برای رخ دادن یک بن بست هر چهار شرط زیر باید برقرار باشند .

1 – انحصار متقابل (Mutual Exclusion)

2 – گرفتن و منتظر ماندن (Hold and wait)

3 – عدم پس گرفتن (انحصاری بودن) (No preemption)

4 – انتظار چرخشی (Circular Wait)

انحصار متقابل بدین معناست که منبع یا منابع در هر لحظه فقط توسط یک پردازش قابل استفاده باشند .

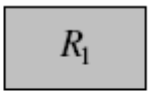
گرفتن و منتظر ماندن یعنی پردازش یک سری منابع مورد نیازش را در اختیار گرفته ، و منتظر منابعی است که در اختیار دیگر پردازش ها است

عدم پس گرفتن بدین معناست که به اجبار نمی توان منبع یا منابع را از پردازش پس گرفت .

انتظار چرخشی یعنی بایستی مجموعه ای از پردازش ها { $P_1,P_2,...,P_n$ } وجود داشته باشد به طوری که P_0 منتظر منبعی از P_1 ، P_1 منتظر منبعی از P_2 و... P_n منتظر منبع از P_0 باشد .

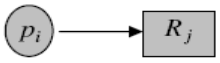
روش توصیف بن بست :

استفاده از گراف تخصیص منابع (Resource Graph) ← G(V,E)

منابع = R= مستطیل 

پردازه= P= دایره 

در گراف تخصیص منابع ، گره ها از نوع منابع یا پردازش هستند، که منابع با مستطیل نمایش داده میشوند و تعداد نمونه های آن با نقطه داخل آن مشخص میشوند و پردازشها با دایره مشخص میشوند .

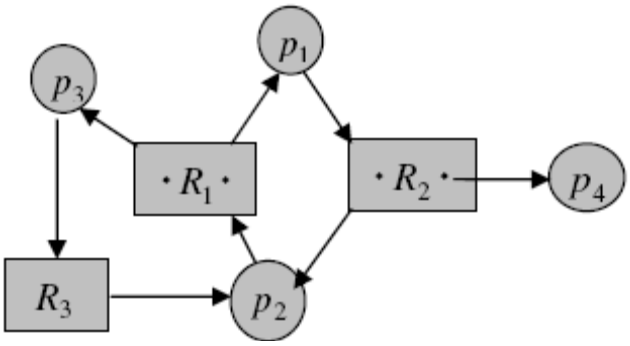
یالهای گراف تخصیص منابع جهت دار میباشند که یا از پردازش به منابه میباشد 

(بدین معنا پردازش Pi منتظر منبع Rj است) یا از منبع به پردازش میباشد 

(بدین معنا که یک نمونه از منبع Rj در اختیار پردازش Pi میباشد)

- ✓ میتوان اثبات کرد که اگر گراف دارای هیچ سیکلی (حلقه یا LOOP) نباشد، هیچ پردازش در سیستم در بنبست نخواهد بود. بعبارت دیگر اگر گراف دارای سیکلی باشد احتمال دارد بن بست وجود داشته باشد. یعنی وجود حلقه در گراف شرط لازم برای بن بست است ، و نه شرط کافی.

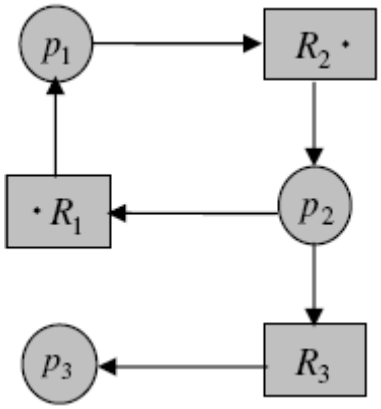
مثال : آیا گراف زیر در بنبست قرار دارد؟

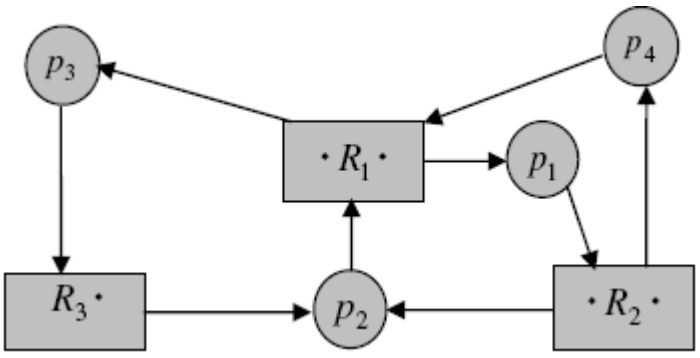


حل : خیر

توضیح : در گراف حلقه وجود دارد ولی سیستم (پردازشها) در بن بست نیستند ، زیرا پردازش P4 به منابع دیگری نیاز ندارد و بنابراین اجرایش را به اتمام رسانده و منبع R2 را رها میسازد ، با رها ساختن منبع R2 بن بست شکسته میشود . منبع R2 به پردازش P1 تخصیص داده میشود ، پردازش P1تمام منابع مورد نیازش را در اختیار دارد ، اجرایش را به پایان رسانده و منابع را آزاد میسازد. در این حال منبع R1 را به پردازه P2 داده ، پردازه P2 نیز با در اختیار داشتن تمام منابع مورد نیازش ، اجرایش را بپایان رسانده و تمام منابع را آزاد میسازد . که یکی از منابع R3 است . منبع R3 به پردازش P3 تخصیص داده شده و این پردازش با در اختیار داشتن منابع اجرایش را بپایان میرساند.

مثال : آیا گرافهای زیر در بن بست قرار دارند؟





حل: بله در هر دو بن بست وجود دارد.

روش های اداره بن بست (Dead lock handling) :

چهار روش برخورد با بن بست وجود دارد :

1. **پیشگیری از بن بست :** برای پیشگیری از بن بست می بایست کاری که یکی از شرایط چهارگانه وقوع بن بست نقص شود . بدین ترتیب هرگز بن بست رخ نخواهد داد .

- ***نقض انحصار متقابل :*** اگر از منابعی به صورت اشتراکی استفاده کنیم بن بست هیچگاه رخ نمی دهد ، نمونه آن فایل های فقط خواندنی هستند که می تواند همزمان توسط چندین پردازش استفاده شوند . سری از منابع که ذاتاً ماهیت غیر اشتراکی دارند قابل استفاده نیست مثل چاپگر .

- ***نقض گرفتن و متنظر ماندن :*** برای نقض این شرط ، باید مانع از ایجاد موقعیتی شد که در آن یک پردازش ، منبعی را در اختیار گیرد و تقاضای منبع دیگری را نماید . رسیدن به این هدف با دو روش امکان پذیر است .

A. همه منابع مورد نیاز پردازش ها در ابتدای شروع اجرای پردازش اگر در دسترس باشند به آن اختصاص داده شوند و گرنه اختصاص داده نشوند به عبارتی یا همه منابع اختصاص داده شوند یا هیچکدام اختصاص داده نشوند .

B. هر پردازش ای که یک سری منابع را در اختیار دارد ، و طلب منابع دیگری می کند می بایست منابع در اختیارش را آزاد سازد ، سپس تمام منابع را با هم تحویل بگیرد .

- ***نقض عدم پس گرفتن :*** جهت تحقق این شرط دو راه حل وجود دارد :

A. یکی این است که اگر پردازش ای درخواست منابعی را داد که آن منابع آزاد نباشند ، تمام منابع در اختیار پردازش ی درخواست پس گرفته شوند .

B. راه حل دیگر آن است که بررسی شود که ایا پردازش های منتظر تمام منابع مورد نیاز پردازش درخواست کننده را دارند یا نه ، که اگر داشته باشند منابع از پردازش های منتظر گرفته شده و به پردازش درخواست کننده انتساب یابد و اگر پردازش های منتظر ، منابع مورد نیاز پردازش ی درخواست کننده را نداشته باشند ، خود پردازش ی درخواست کننده باید منتظر بماند و ممکن است در حین انتظار منابع در اختیارش نیز از وی گرفته شوند و به پردازش های دیگری انتساب یابند .

- ***نقض انتظار چرخشی :*** جهت نقض انتظار چرخشی می بایست منابع را شماره گذاری کرد و به طوری که هر پردازش بتواند منابع را در جهت صعودی شماره هایشان درخواست کند . به عنوان مثال اگر پردازش

ای منبع شماره 3 را در اختیار داشته باشد ، منبع شماره ی 1 را نمی تواند درخواست بکند ، ولی می تواند منبع شماره ی 5 را درخواست کند .

عیب روش پیشگیری از بن بست ، بهره وری پایین منابع و کاهش توان عملیاتی سیستم می باشد .

2. اجتناب از بن بست :

در این روش با توجه به اطلاعاتی نظیر حداکثر نیاز پردازش ها به منابع و منابع تخیص یافته به پردازش ها و موجودی ، وقتی پردازش ای درخواست یک سری منابع را داشته باشد ، اگر منابع موجود باشند ، بررسی می کنیم که آیا با اجابت این درخواست سیستم به حالت امن میرود یا نه . اگر سیستم به حالت امن برود درخواست تخیص داده می شود وگرنه از درخواست اجتناب می شود .

حالت امن حالتی است که پردازش ها میتوانند به ترتیب خاص منابع مورد نیازشان را گرفته و اجرایشان را با موفقیت پشت سر بگذرانند .

الگوریتم بانکداران :

این الگوریتم اولین بار توسط دیجسترا ارائه شد، و به نام الگوریتم بانکدار معروف گردید ، چرا که این الگوریتم شبیه ، رفتار یک بانکدار شهر کوچک با مشتریانش طراحی شده است ، یک بانکدار هرگز تمام سرمایه خودش را به مشتریان تخصیص نمیدهد و طوری عمل میکند که بتواند کلیه نیازهای مشتریانش را برآورده کند.

افراد	اعتبار استفاده شده	حداکثر اعتبار
Andy	0	6
Barbara	0	4
Marvin	0	5
Suzanne	0	7

واحدهای در دسترس=10

شکل (الف)

افراد	اعتبار استفاده شده	حداکثر اعتبار
Andy	1	6
Barbara	1	5
Marvin	2	4
Suzanne	4	7

واحدهای در دسترس=2

شکل (ب)

افراد	اعتبار استفاده شده	حداکثر اعتبار
Andy	1	6
Barbara	2	5
Marvin	2	4
Suzanne	4	7

واحدهای در دسترس=1

شکل (ج)

در شکل الف چهار مشتری داریم که به هر یک از آنها مقدار مشخصی اعتبار اعطاء شده است . بانکدار میداند که همه مشتریانش حداکثر اعتبار در نظر گرفته شده را نیاز نخواهند داشت ، بنابراین بجای 22 واحد ، 10 واحد را

برای ارائه خدمات به آنها ذخیره کرده است . (در این مثال مشتریان همهن پردازش ها هستند و بانکدار، سیستم عامل میباشد)

مشتریها هر از چند گاهی درخواست وام میکنند، در لحظه معینی وضعیت به صورت شکل ب خواهد بود. در این شکل فهرستی از مشتریها وجود دارد و در کنار آن اعدادی ذکر شده که نشان دهنده مقدار پولی است که قبلا به آن شخص وام داده شده است (نوارگردانهایی که قبلا واگذار شده است). و نیز اعدادی که بیانگر حداکثر اعتبار ممکن میباشد (حداکثر تعداد نوار گردانهایی که بعدا لازم خواهد شد). این اطلاعات وضعیت سیستم را با توجه به تخصیص منابع بیان میکنند.

یک وضعیت زمانی، امن گفته میشود که ترتیبی از دیگر وضعیتها وجود داشته باشد که منجر به این مسئله شود که همه مشتری ها به اندازه حداکثر اعتبارشان وام دریافت نمایندو بروند.ضعیت شکل ب امن میباشد. زیرا با دو واحد باقیمانده ، بانکدار میتواند هر درخواستی را به جز درخواست "marvin" به تاخیر اندازد، بنابراین "marvin" فرصت خواهد داشت که کار خود را به پایان برساند و هر چهار منبع در اختیار خود را رها سازد، با داشتن 4 واحد ، بانکدار میتواندااجازه دهد که "Suzanne" یا "Barbara" احد های مورد نیاز خود را دریافت نمایند و ادامه به همین صورت خواهد بود.

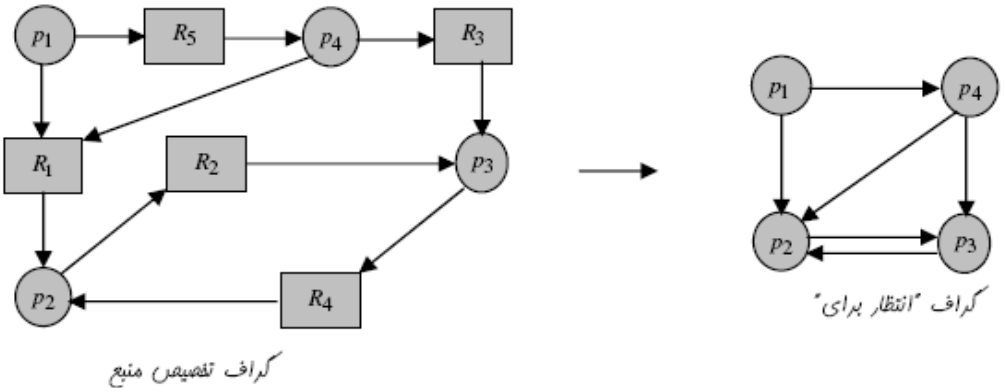
در صورتی عدم یک الگوریتم مشتریان ناگهان حداکثر وامشان را درخواست میکردند، و بانکدار نمیتوانست هیچ کدام از آنها را راضی نگهدارد. یعنی به یک بن بست میرسیدیم. یک وضعیت ناامن الزاما به یک بن بست منتهی نمیشود ، زیرا ممکن است یک مشتری نیازی به حداکثر اعتبار موجود نداشته باشد، ولی بانکدار نمیتواند روی این وضعیت حساب باز کند.

بنابراین الگوریتم بانکدار به این قرار است که هر دو درخواست را به محض اینکه اتفاق افتاد در نظر بگیریم و ببینیم که آیا پذیرش آن منتهی به وضعیت امن خواهد شد یا خیر . اگر وضعیت امن بود، آن را می پذیریم و در غیر اینصورت آنرا به تاخیر می اندازیم. برای فهمیدن اینکه وضعیت امن است یا خیر بانکدار بررسی میکند تا ببیند که آیا منبع کافی برای راضی نمودن آن مشتری که تفاضل وام دریافتی فعلی او از حداکثر کمتر از سایرین است ، را در اختیار دارد یا خیر ، اگر چنین بود، وامهای آن مشتری ، بازپرداخت شده در نظر گرفته میشود و در میان سایرین ، دوباره آن مشتری که اکنون به حداکصر اعتبارش نزدیکتر از بقیه می باشد ، کنترل میشود و به همین ترتیب ادامه پیدا می یابد. اگر عاقبت همه وامها بتواند بازپرداخت شوند، وضعیت بوجود آمده در صورت پذیرش درخواست قبلی امن خواهد بود و آن درخواست میتواند پذیرفته شود.

3. **روش تشخیص بن بست و بازیافت سیستم :**

در این روش از روی گراف تخصیص منبع، گراف انتظار(wait for graph) را بدست می آوریم . برای بدست آوردن گراف انتظار ، گره های منبع را از گراف تخصیص حذف کرده و کمان های مناسبی را با هم ترکیب میکنیم .

مثال : فرض کنید گراف تخصیص منابع به شکل زیر است اولا گراف انتظار را رسم کنید ثانیا مشخص کنید که سیستم دچار بن بست شده است یا خیر؟



چون حلقه وجود دارد سیستم دچار بن بست شده است.

یک پیکان از پردازش Pi به Pj در گراف انتظار ، نمایانگر این است که پردازش Pi در انتظار پردازش Pj است تا منبعی را که Pi به آن نیاز دارد ، آزاد کند. سیستم عامل ، گراف انتظار را نگهداری کرده و مرتبا آن را چک میکند، اگر در گراف انتظار حلقه وجود داشته باشد، آنگاه سیستم به بن بست رسیده است .

4. **روش صرف نظر کردن از بن بست (الگوریتم استریخ Ostrich) :** در این روش در واقع هیچ عملی در مقابل بن بست انجام داده نمی شود . در صورتی که بن بست منجر به از کار افتادن سیستم شود (Hang) آنگاه سیستم به صورت دستی ریست (reset) می شود .

جالب است که بدانید در اکثر سیستم عامل های امروزی مثل unix از همین روش چهارم استفاده می شود ، چرا که در این سیستم ها بن بست به ندرت رخ می دهد (مثلاً سالی یک بار) لذا ارزنتر آن است که به جای روش های پر هزینه پیشگیری ، اجتناب و آشکار سازی کلاً از این مشکل چشم پوشی کنیم .

ترکیب روش ها در اداره بن بست

در عمل هر یک از روش های اداره بن بست (پیشگیری ، اجتناب ، کشف) به تنهائی برای تمام انواع منابع مناسب نمی باشد ، یک شیوه ترکیبی برای دسته های مختلف منابع مثلاً می تواند به صورت زیر باشد .

- منابع داخلی سیستم مثل بلوک کنترل پردازش : پیشگیری از طریق ترتیب منابع

- منابع کار (گرداننده های دیسک ، نوار ، چاپگر و ...) : اجتناب ار بن بست ، چون حداکثر نیاز از قبل مشخص است .

- حافظه اصلی : پیش گیری از طریق پس دادن می تواند انجام پذیرد ، چرا که به راحتی می توان حافظه اصلی را از پردازش ها پس گرفت ؛ زیرا به محض کمبود حافظه یکسری از پردازش ها به حافظه جانبی منتقل می شوند .

- حافظه جا به جا پذیر (پیشیبان) : تخصیص از پیش ، می تواند انجام پذیرد چرا که حداکثر نیازهای ذخیره سازی از قبل می تواند مشخص باشد .

- منابع پردازش : از طریق اجتناب

مدیریت حافظه (قطعه بندی و صفحه بندی) :

یکی از مولفه های سیستم عامل مدیریت حافظه است .

وظایف مدیر حافظه :

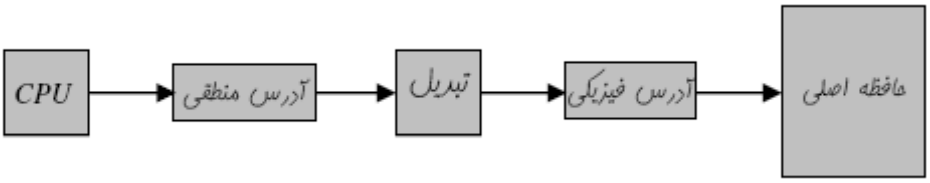
- اداره کردن سلسله مراتب حافظه
- جلوگیری از تداخل برنامه های موجود در حافظه (به خصوص در محیط های چند برنامه گی)
- مدیریت حافظه مجازی

آدرس منطقی : آدرسی است که برنامه نویس در برنامه صادر میکند یا آدرسی که توسط CPU تولید می شود .

آدرس فیزیکی: آدرس مشاهده شده توسط واحد حافظه (یعنی آنچه که در جیستر آدرس حافظه ، بار می شود) را آدرس فیزیکی می نامند .

تبدیل آدرس منطقی به آدرس فیزیکی :

به عمل تبدیل آدرس منطقی به آدرس فیزیکی نگاشت آدرس (maping) گویند .



انواع انقیاد آدرس :

1. ***زمان کامپایل*** *:* اگر در موقع کامپایل معلوم باشد که برنامه در کجای حافظه قرار خواهد گرفت ، در این صورت کد مطلق میتواند تولید شود ، یعنی آدرسهای ذکر شده در برنامه هنگام بار شدن یا هنگام اجرا تغییر نخواهد کرد و تصویر آینه وار برنامه در دیسک عینا به حافظه آورده شده و اجرا میگردد. مثلا با آدرس 100 ذکر شده در برنامه همان آدرس 100 مطلق حافظه RAM میباشد.
2. ***زمان بارکردن:*** اگر در زمان کامپایل معلوم نباشد که برنامه در کجای حافظه قرار خواهد گرفت ، آنگاه کامپایلر بایستی کد قابل جابجائی تولید کند .
3. ***زمان اجرا*** **:** اگر پردازش در حین زمان اجرایش بتواند در حافظه جابجا شود ، آنگاه پیوند دادن بایستی تا زمان اجرا به تاخیر انداخته شود. برای این حالت نیاز به سخت افزار خاصی وجود دارد.

روشهای مدیریت حافظه :

1. تک برنامگی ساده :

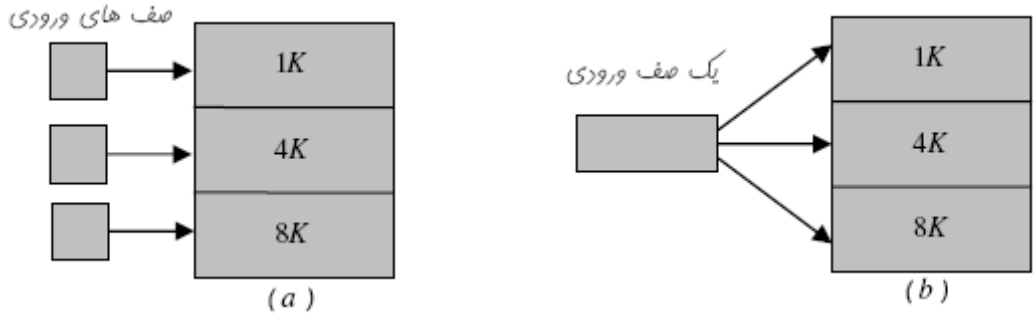
در این روش در هر لحظه فقط یک برنامه در حافظه اصلی قرار دارد، و برنامه ای که می بایست اجرا شود ، نباید اندازه اش از حافظه اصلی بیشتر باشد اگر حافظه RAM به اندازه کافی در دسترس نباشد، برنامه اجرا نمیشود.سیستم عامل DOS اولیه این گونه بوده است .(شکل زیر)

2. یک برنامگی با سیستم :

در این سیستم مدیریت ، پردازش میتواند بزرگتر از حافظه اصلی باشد . در این شیوه برنامه به بخشهای مختلفی تقسیم شده و تنها آن داده ها و دستورالعملهائی را در حافظه قرار نی دهیم که در هر زمان مفروض مورد نیاز هستند و بقیه بخش ها در دیسک باقی می مانند . هنگامی که به بخش دیگری از آن برنامه نیاز داریم ، قسمتی که مورد نیاز نیست از خارج شده و بخش مورد نیاز به حافظه آورده می شود.اسن تکنیک به حمایت سخت افزاری نیاز ندارد و خود برنامه نویس می بایست آن را در برنامه پیاده سازی کند.

3. چند برنامگی با بخش بندی ثابت حافظه :

ساده ترین روش چند برنامه گی این است که خافظه را به N قسمت تقسیم کنیم، اندازه هر قسمت می تواند با بخش های دیگر متفاوت باشد . این کار میتواند در هنگام شروع کار سیستم توسط سیستم عامل یا به صورت دستی توسط اپراتور انجام شود. وقتی یک کار وارد میشود در یک صف ورودی قرار میگیرد تا در کوچکترین بخش که مناسب آن است قرار داده شود. البته ممکن است آن بخش دقیقا هم اندازه برنامه نبوده و بدین ترتیب مقداری از فضای آن از بین برود . در این روش میتوان برای هر پارتیشن از یک صف مجزا استفاده کرد (شکل a) و یا اینکه فقط یک صف برای تمام پارتیشن ها داشت (شکل b).



4. **چند برنامگی با جابجائی (Swapping)**

در این روش در هر لحظه میتوان چندین پردازش داشت ، ولی اگر پردازش میخواد به حافظه اصلی بیاید، و حافشظه خالی نباشد ، بجای یکی از پردازش های موجود در حافظه اصلی قرار میگیرد، و پردازشی که در حافظه اصلی بود به دیسک منتقل میشود. اشکال این روش این است که به دلیل نقل و انتقال پردازش مابین حافظه اصلی و دیسک زمانی زیادی طول میکشد.

5. چند برنامگی بصورت تخصیص همجوار

در این روش حافظه از قبل به اندازه های ثابت قسیم نمیشود بلکه پردازشهائی که میبایست به حافظه بیایند مطابق الگوریتم هائی یکی از فضا های آزاد حافظه را پیدا کرده و بطور کامل در آن قسمت قرار میگیرد.

تکه تکه شدن (پارگی):

به حفره های حافظه که ظرفیت مجموع آنها زیاد بوده ولی چون از همدیگر فاصله دارند قابل استفاده نیستند را پارگی خارجی گوئیم که یک راه برای از بین بردن آنها فشرده سازی می باشد (کاری میکنیم که فضاهای آزاد در کنار هم قرار گیرند یعنی برنامه ها را جابجا کنیم)، فشرده سازی عملی زمانبر است و از طرفی فقط کدهایی را میتوان جابجا کرد که جابجا پذیر باشند.

الگوریتم های انتخاب جا (انباره) :

1. اولین مناسب (frist fit)

وقتی پردازش ای می خواهد به حافظه load شود ابتدای لیست فضای آزاد را نگاه کرده و اولین فضای آزادی که اندازه اش بزرگتر یا مساوی اندازه پردازش باشد انتخاب شده و پردازش در آن محل قرار می گیرد .
مشکل تراکم پردازش ها در ابتدای حافظه است که روش بعدی سعی می کند این مشکل را برطرف کند .

2. مناسب بعدی (Next fit)

این روش مانند frist fit است با این تفاوت که جستجو از محلی در لیست آغاز می شود که آخرین بار تخصیص از آن محل صورت گرفته است . بدین ترتیب یکنواختی توزیع برنامه در سطح حافظه نسبت به روش قبلی بیشتر خواهد شد .

3. بهترین مناسب (Best fit)

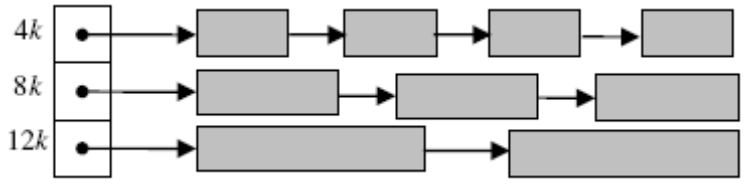
در این روش کل لیست فضای آزاد جستجو شده و کوچکترین حفره که به اندازه کافی بزرگ است به پردازش تخصیص داد می شود . این روش باعث می شود که کوچکترین حفره بر اثر تخصیص باقی بماند . با این روش فضاهای بزرگتر برای تقاضای بیشتر حفظ می شوند . از آنجا که تمام لیست بلاک های آزاد باید بررسی شود ، این تکنیک قدری زمانبر است .

4. بدترین مناسب (worst – fit) :

در این الگوریتم بزرگترین حفره انتخاب شده و پردازش در آن قرار می کرد . دلیل انتخاب بزرگترین حفره این است که از فضای باقی مانده دیگر پردازش ها می توانند استفاده کنند . ایراد این تکنیک این است که امکان دارد ، تقاضاهای که ناحیه بزرگی می خواهند ، دیگر نتوانند برآورده شوند چرا که بلاک های بزرگ زودتر تخصیص یافته و کوچک می شوند .

5. سریعترین مناسب (Quick Fit) :

در این الگوریتم لیستی از اندازه پردازش های متداول تهیه می شود و آرایه ای با n خانه در نظر گرفته می شود که هر خانه این آرایه شامل یک اشاره گربه ابتدای لیست یک فضای خالی به اندازه متداول است به عنوان مثال فضاهای متداول می توانند $12K,8K,4K,2K$ و ... باشند که برای هر کدام یک خانه آرایه در نظر گرفته می شود . عیب این روش این است که اگر پروسسی خاتمه یابد باید فضای آزاد شده آن به لیست مناسب اضافه شود که این کار زمانبر می باشد .



6. الگوریتم رفاقتی (Buddy) :

در این روش حفره ها (فضاهای خالی) به صورت توان های 2 در نظر گرفته می شود . به عنوان مثال حفره هائی به اندازه $32K,16K,8K,4K,2K,1K$ ... و برای هر گروه یک لیست جداگانه در نظر گرفته می شود . بدین ترتیب جهت تخصیص یک بلاک تنها باید بلاک مورد نظر را از لیست مناسب خارج کرد . پس از تخصیص اگر فضای باقی مانده آن بلاک ، توانی از 2 باشد در لیست مربوطه اش قرار می گیرد و در غیر این صورت به چنیدن بخش که اندازه هر کدام توانی از 2 می باشد تقسیم می شود . از طرف دیگر در این روش بلوک های کنار هم می توانند با هم ترکیب شده و بخش بزرگتری را پدید بیاورند .

مثال : با قسمت هائدی از حافظه با اندازه هائی $600k,300k,200k,500k,100k$ هر یک از روش های اولین جای مناسب ، بهترین جای مناسب و بدترین جای مناسب پردازش هائی با اندازه $426k,112k,417,121k$ را چگونه در حافظه قرار می دهند و کدام روش از حافظه به طور بهینه استفاده می کند . ترتیب ورود پردازش ها را یک بار از دست به چپ و یک بار چپ به راست بگیرید .

$H: 100k\ ,\ 500k\ ,\ 200k\ ,\ 300\ ,\ 600k$
$P: 212k\ ,\ 417\ ,\ 112k\ ,\ 426k$

الف . اولین مناسب : در حالت از چپ به راست پردازش 426 باید منتظر بماند .

$100k\ ,\ 176k\ ,\ 200k\ ,\ 300k\ ,\ 183k\ ,$ لیست فضای آزاد، ترتیب پردازه ها از چپ به راست
$100k\ ,\ 74k\ ,\ 88k\ ,\ 88k\ ,\ 183k$ لیست فضای آزاد، ترتیب پردازه ها از راست به چپ

ب . بهترین مناسب

$100\ k\ ,\ 83\ k\ ,\ 88\ k\ ,\ 88\ k\ ,\ 174\ k\ ,$ لیست فضای آزاد، ترتیب پردازه ها از چپ به راست
$100\ k\ ,\ 74\ k\ ,\ 88\ k\ ,\ 88\ k\ ,\ 183\ k$ لیست فضای آزاد، ترتیب پردازه ها از راست به چپ

ب . بدترین مناسب

$100\ k\ ,\ 388\ k\ ,\ 200\ k\ ,\ 88\ k\ ,\ 174\ k\ ,$ لیست فضای آزاد، ترتیب پردازه ها از چپ به راست
$100\ k\ ,\ 83k\ ,\ 200\ k\ ,\ 188\ k\ ,\ 388\ k$ لیست فضای آزاد، ترتیب پردازه ها از راست به چپ

روش بهترین مناسب از حافظه به صورت بهینه استفاده می کند . زیرا برای تمام پردازش ها فضای لازم را پیدا می کند و پارگی خارجی در آن حداقل است . اندازه $35k,20k,30k,20k$ به ترتیب از راست به چپ ذکر سده داده شود و از روش Next Fit استفاده شود و تخصیص از اول حافظه شروع شود ، وضعیت حافظه را بعد از این تخصیص ها مشخص کنید .

شروع $\rightarrow 40k\ ,\ 25k\ ,\ 45k\ ,\ 50k\ ,\ 60k\ ,\ 40k$

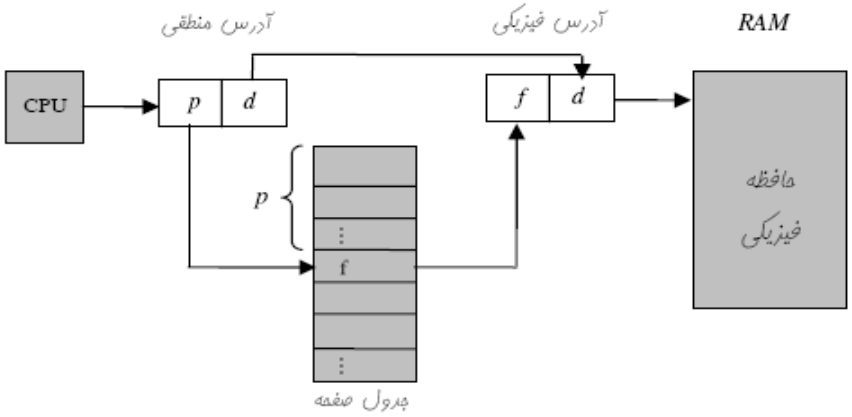
جواب: $20k\ ,\ 25k\ ,\ 15k\ ,\ 30k\ ,\ 25k\ ,\ 40k$

صفحه بندی (Paging) :

در روش صفحه بندی برنامه کاربر (فضای آدرس منطقی) به بخش هائی به اندازه ثابت به نام صفحه (Page) تقسیم می شود و حافظه فیزیکی به قسمت هائی به نام قاب (Frame) تقسیم می شود که اندازه هر فریم برابر اندازه Page می باشد . صفحه بندی این امکان را می دهد که قسمت های یک برنامه (Page های هر برنامه) در حافظه پراکنده باشند .

نحوه تبدیل آدرس منطقی به آدرس فیزیکی :

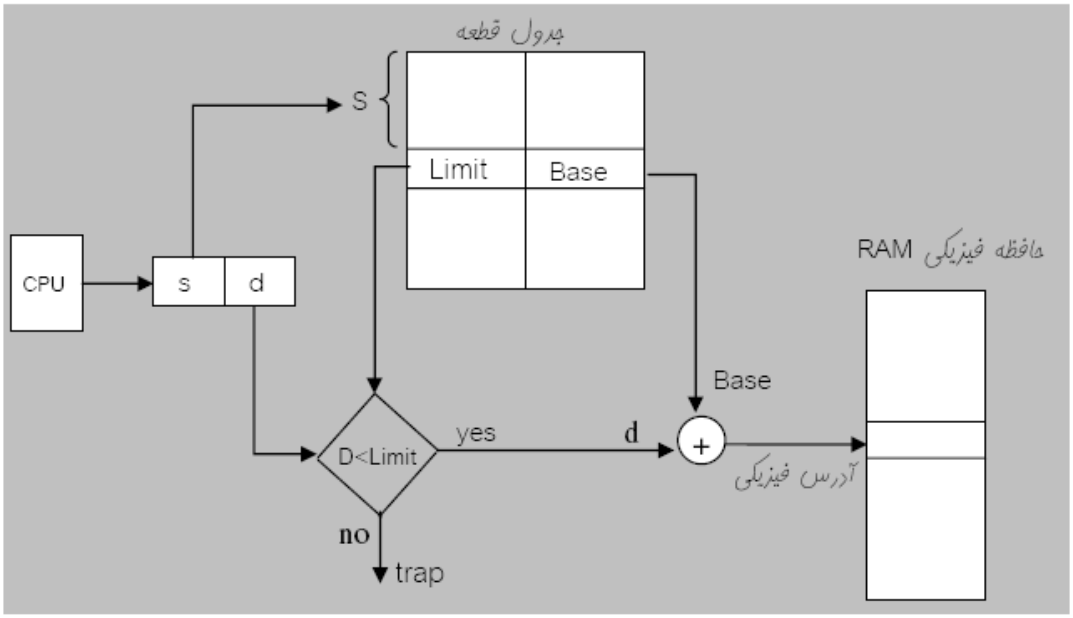
در این تکنیک هر آدرس تولید شده توسط CPU (یعنی آدرس منطقی) از دو بخش شماره صفحه (P) و آفست صفحه (d) تشکیل شده است . شماره صفحه به عنوان اندیس جدول صفحه (page table) استفاده می گردد . جدول صفحه شامل آدرس مبنای هر صفحه در آدرس فیزیکی RAM است . این آدرس مبنا با آدرس افست منطقی ترکیب شده و آدرس فیزیکی نهائی را تشکیل می دهد . شکل زیر این موضوع را نشان می دهد .



قطعه بندی (segmentation)

حافظه اصلی یا فیزیکی به صورت یک آرایه خطی از بابت ها یم باشد به عبارتی هر خانه یک آدرس دارد ، در روش صفحه بندی پردازش ها بر اساس محدودیت فیزیکی یعنی اندازه هر صفحه تقسیم می شوند ولی در قطعه بندی خود کاربر می تواند برنامه اش را به صورت منطقی تقسیم کند که به هر یک از این قسمت ها یک segment گویند هر قطعه یک آدرس شروع و یک طول دارد ، آدرسی که کاربر در سطح منطقی می دهد شامل دو جزء است یکی شماره قطعه و دیگری فاصله (آفست) خانه مورد نظر از اول آن قطعه است ، آدرس دو بعدی استفاده شده توسط کاربر در سطح منطقی می بایست توسط یک نگاشت به آدرس یک بعدی فیزیکی تبدیل شود . این نگاشت به وسیله جدول قطعه انجام می پذیرد ، جدول قطعه از دو ستون اصلی تشکیل شده است ، یکی

آدرس پایه قطعه (Base) و دیگری طول یا حد قطعه (Limit) . قسمت Base حاوی آدرس فیزیکی در RAM می باشد که قطعه از آنجا شروع می شود ، شکل زیر نحوه تبدیل آدرس منطقی را به آدرس فیزیکی در این سیستم نشان می دهد .



آدرس منطقی از دو جزء شماره قطعه (s) و افست درون آن قطعه (d) تشکیل یافته است .

یکی از مزایای قطعه بندی اشتراک است .

در صفحه بندی اندازه صفحات برابر است ولی در قطعه بندی لزومی ندارد که اندازه صفحات یکی باشد .

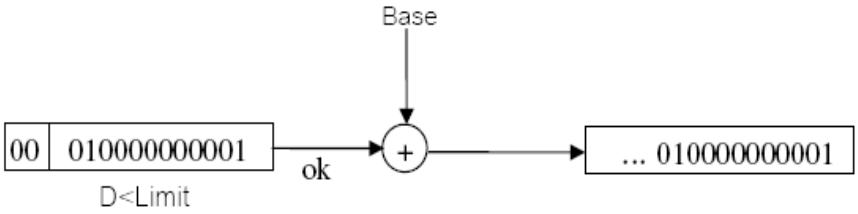
مثال . فرض کنید در یک شبکه کامپیوتری چندین کاربر به صورت همزمان نیاز به اجرای برنامه word دارند به جای این که هر برنامه word که برای همه مشترک است برای هر کاربر به صورت مجزا در حافظه load شود ، یک بار در حافظه load شده و تمامی کاربران به صورت اشتراکی از آن استفاده می کنند و هر کاربر قطعه خاص خود را دارد .

مثال . فرض کنید پردازش ای دارای سه قطعه می باشد به طوریکه قطعه اول $2K$ ، قطعه دوم $3K$ ، و قطعه سوم $4K$ می باشد مطلوب است .

الف . تعیین ورودیهای جدول قطعه

	base	Limit	V
00	0000	2×2^{10}	1
01	3000	3×2^{10}	1
10	2000	4×2^{10}	1
11			0

ب . اگر برنامه نویس در قطعه دستور push با آدرس 1025 را داشته باشد آدرس فیزیکی متناظر با این آدرس را بیابید .



نکته : اگر برنامه نویس دستور pop خانه 2048 را در قطعه ی صفر بدهد چون آدرس صادر شده در قطعه صفر وجود ندارد بنابراین وقفه رخ می دهد .

ترکیب قطعه بندی و صفحه بندی

هم صفحه بندی و هم قطعه بندی نقاط قوت خصوص به خود را دارند برای ترکیب نقاط قوت هر دو ، این دو روش را ترکیب کرده و با هم استفاده می کنیم ، در یک سیستم ترکیبی صفحه بندی / قطعه بندی فضای آدرس کاربر تحت نظر برنامه ساز به تعدادی قطعه تقسیم می شود ، هر قطعه به نوبه خود به تعدادی صفحه به اندازه ثابت تقسیم

می گردد . به ازای هر فرایند یک جدول قطعه و به ازای هر قطعه یک جدول صفحه ایجاد می گردد ، تعداد درایه های هر جدول صفحه بستگی به اندازه قطعه مربوطه دارد . همانند صفحه بندی آخرین صفحه هر قطعه معمولاً پر نمی شود و به ازای هر قطعه به طور میانگین به اندازه نصف صفحه تکه تکه شدن داخلی داریم . از دید برنامه ساز ، آدرس منطقی همچنان شامل شماره قطعه و انحراف در قطعه است . از دید سیستم این انحراف در قطعه ، به صورت یک شماره صفحه و انحراف در صفحه دیده می شود .

هنگامی که فرایندی در حال اجراست یک ثبات آدرس شروع جدول قطعه آن فرایند را نگه می دارد . پردازنده از شماره قطعه ای که در آدرس منطقی است به عنوان شاخص به جدول قطعه استفاده کرده و جدول صفحه را برای قطعه استفاده کرده و جدول صفحه را برای قطعه مذبور پیدا می کند ، سپس بخش شماره صفحه آدرس منطقی به عنوان شاخص جدول صفحه به کار رفته و شماره قاب مربوطه بدست می آید . این شماره قاب با بخش انحراف آدرس منطقی ترکیب شده و آدرس فیزیکی مورد نظر نتیجه می شود . شکل زیر نحوه ترجمه آدرس در یک سیستم قطعه بندی / صفحه بندی را نشان می دهد

مهمترین مزیت صفحه بندی :

پارگی خارجی ندارد

در قطعه بندی پارگی خارجی داریم زیرا اندازه قطعات مساوی نیست و قطعه ای بلا استفاده ای وجود دارد که پراکنده بودن و همجوار نیستند بنابراین نمی توان از آنها برای یک قطعه استفاده کرد .

مهمترین مزیت قطعه بندی

اشتراک

اشتراک در صفحه بندی نسبت به قطعه بندی ناچیز است و یا اصلاً وجود ندارد .

حافظه مجازی :

تکنیکی است که اجازه می دهد بدون این که کل برنامه در حافظه اصلی قرار گیرد اجرا شود . مزیت مهم این روش آن است که اجازه می دهد برنامه بزرگتر از حافظه اصلی باشد . علاوه بر این در این روش کاربر حافظه را به صورت آرایه ای فوق العاده بزرگ و یکنواخت می بیند . به عبارتی کاربر (برنامه نویس) خود را درگیر حافظه فیزیکی نمی کند و فرض می کند که بی نهایت حافظه در اختیار دارد و برنامه نویسی ساده می شود . مزیت دیگر این است که چون فقط صفحات مورد نیاز از دیسک به حافظه اصلی منتقل می شوند بنابراین زمان کمتری صرف عمل I/O می شود از این رو زمان پردازش نیز سریعتر می شود .

در هر مراجعه به حافظه اعمال زیر رخ می دهد

در روش مدیریت حافظه مجازی به آدرس منطقی آدرس مجازی می گویند اگر بیت 7 یک باشد بدین معناست که page مورد نیاز در حافظه اصلی موجود است اگر صفر باشد دو برداشت از آن وجود دارد .

1 – چنین page ی اصلاً وجود ندارد یعنی دسترسی غیر مجاز است به عنوان مثال پردازش شما دارایث سه page است ولی شما مراجعه به page چهارم را می دهید که این دسترسی غیر مجاز است .

2 – چنین page ی وجود دارد ولی در حافظه اصلی نیست .