

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

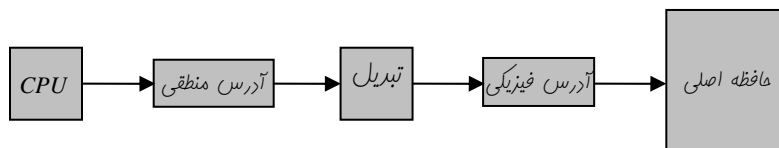
آدرس منطقی: آدرسی است که برنامه نویس در برنامه صادر میکند یا آدرسی که توسط CPU تولید میشود. مثلا آدرس منطقی 100 در کد

`Mov Al, [100]`

آدرس فیزیکی: آدرس مشاهده شده توسط وایر حافظه (یعنی آنچه که در رجیستر آدرس حافظه بار میشود) را آدرس فیزیکی می نامند. آدرس مجازی: در حالتی که پیوند آدرس های حافظه در زمان اجرا باشد، به آدرس منطقی، آدرس مجازی (Virtual Addresss) میگویند.

تبدیل آدرس منطقی به آدرس فیزیکی:

به عمل تبدیل آدرس منطقی به آدرس فیزیکی نگاشت آدرس (mapping) 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 آدرس های ذکر شده در برنامه هنگام بارشدن و یا هنگام اجرا تغییر نخواهد کرد و تصویر آینه وار برنامه در دیسک عینا به به حافظه آورده شده و اجرا می گردد. مثلا آدرس 100 ذکر شده در برنامه همان آدرس 100 مطلق حافظه RAM می باشد. برنامه های com تحت سیستم عامل DOS اینگونه هستند.
- 2- **زمان بار کردن:** اگر در زمان کامپایل معلوم نباشد که برنامه در کجای حافظه قرار خواهد گرفت، آنگاه کامپایلر بایستی کد قابل جابه جایی (Relocatable) تولید کند. به عنوان مثال اگر در برنامه ای دستور `Mov Al [100]` را داشته باشیم و برنامه حاوی این دستور در زمان بار کردن در محل 300 حافظه قرار گیرد (شروع آدرس 300 باشد) در این صورت آدرس فیزیکی معادل آدرس 100 برابر 400 میشود.
- 3- **زمان اجرا:** اگر پردازش در حین زمان اجرا بتواند در حافظه جابه جا شود، آنگاه پیوند دادن بایستی تا زمان اجرا به تأخیر انداخته شود. برای این حالت نیاز به سخت افزار خاصی وجود دارد.

□ هم برای نگاشت زمان کامپایل و هم برای زمان نگاشت بار کردن مدیر حافظه ملزم به پشتیبانی سخت افزاری نیست ولی برای نگاشت زمان اجرا نیاز به پشتیبانی سخت افزاری است.

روش های مدیریت حافظه:

1- تک برنامه گوی سازه:

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 نمی شود. سیستم عامل DOS اولیه این گونه بوده است.

برنامه های ROM
برنامه های کاربر
سیستم عامل

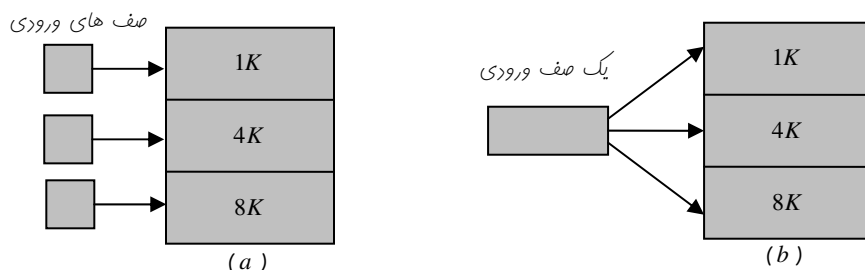
طرح حافظه در این سیستم عامل اولیه به این شکل بوده است

2- یک برنامه گنی با سیستم (overlay)

در این سیستم مدیریت، پردازش می تواند بزرگتر از حافظه اصلی باشد. در شیوه overlay (بایگلاشت) برنامه به بخش های مختلفی تقسیم شده و تنها آن داده ها و دستورالعمل هائی را در حافظه قرار می 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 حافظه خارج شده و بخش مورد نیاز به حافظه آورده می شود. □ overlay به حمایت سفت افزاری ویژه ای نیاز ندارد. این تکنیک را خود برنامه نویس می بایست در برنامه پیاده سازی کند.

3- چند برنامه گنی با بخش بندی ثابت حافظه:

ساده ترین روش چند برنامه گنی این است که حافظه را به N قسمت تقسیم کنیم، اندازه هر قسمت می تواند با بخش های دیگر متفاوت باشد. این کار می تواند در هنگام شروع کار سیستم توسط سیستم عامل یا به صورت دستی توسط اپراتور انجام شود. وقتی یک کار وارد می شود در یک صف ورودی قرار می گیرد تا در کوچکترین بخش که مناسب آن است قرار داده شود. البته ممکن است آن بخش دقیقاً هم اندازه برنامه نبوده و بدین ترتیب مقداری از فضای آن از بین برود. در این روش می توان 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 برنامه گنی با جابه جایی (Swapping) - مبارله

در این روش در هر لحظه میتوان چندین پردازش داشت، ولی اگر پردازش ای می خواهد به حافظه اصلی بیاید، و حافظه خالی نباشد، به جای یکی از پردازش های موجود در حافظه اصلی قرار می گیرد، و پردازش ای که در حافظه اصلی بود به دیسک منتقل میشود. اشکالی که در این روش وجود دارد این است که به دلیل نقل انتقال پردازش مابین حافظه اصلی و دیسک زمانی طول می کشد.

$$CPU \text{ کارایی} = \frac{\text{زمان اجرای پردازش}}{\text{زمان جابه جایی} + \text{زمان اجرای پردازش}}$$

کل زمان طی شده

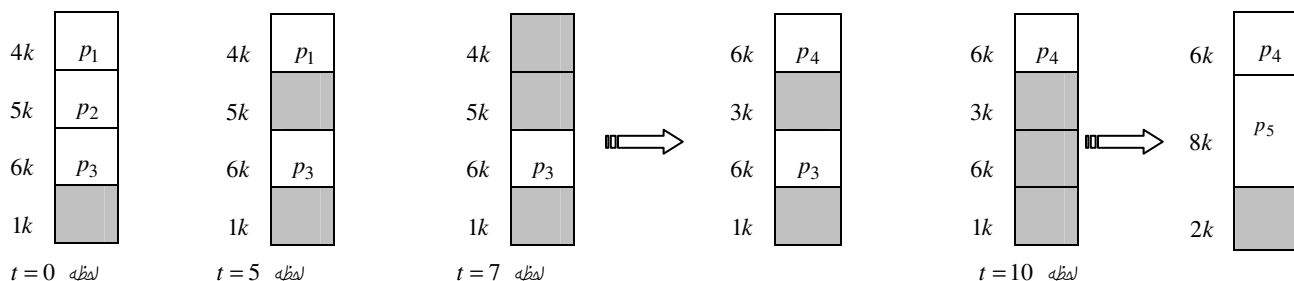
5- چند برنامه گنی به صورت تفصیص همجواری:

در این روش حافظه از قبل به اندازه های ثابت تقسیم نمی شود بلکه پردازش هائی که می بایست به حافظه آیند مطابق الگوریتم هائی یکی از فضا های آزاد حافظه را پیدا کرده و به طور کامل در آن قسمت قرار می گیرد.

زمان اجرا	مقدار حافظه مورد نیاز	job
3	4k	j_1
2	5k	j_2
4	6k	j_3
3	6k	j_4
3	8k	j_5

مثال. فرض کنید اندازه حافظه ای 16k باشد اگر کار های زیر با توجه به الگوریتم FCFS توسط زمان بند بلند مدت جهت تبدیل شدن به پردازش ها انتفا شوند و در عین اجرای پردازش ها از زمانبندی RR با کوانتوم زمانی 1 استفاده شود. وضعیت حافظه را بعد از هر Load شدن برنامه جدید به حافظه مشخص نموده و زمان Load شدن پردازش ها را معین کنید (پردازش وارد شده به انتهای صف می رود).

ابتدا برنامه های p_1, p_2, p_3 در حافظه بار شده و شروع به اجرا میشوند. توجه کنید به علت نبود حافظه کافی



p_1	p_2	p_3	p_1		p_2	p_3	p_1	p_3	p_4	p_3	p_4	p_5	p_4	p_5	p_5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

پارگی:

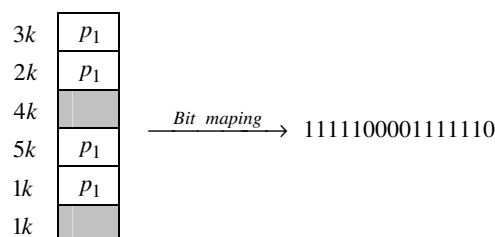
به فهره های حافظه که ظرفیت مجموع آنها زیاده ولی چون از هم دیگر فاصله دارند قابل استفاده نیستند را پارگی خارجی گوئیم که یک راه برای از بین بردن آنها فشرده سازی می باشد (کاری می کنیم که فضا های آزاد در کنار هم قرار گیرند یعنی برنامه ها را جابه جا کنیم)، فشرده سازی عملی زمانبر هست و از طرفی کدهائی را می توان جابه جا کرد که جابه جایی (Relocatable) باشند.

□ اگر زمانی پردازه ای مانند z_4 بخواهد وارد حافظه اصلی شود یک فضای $7k$ خالی در ابتدای حافظه و یک فضای خالی $6.5k$ در محل دیگر حافظه باشد، در کدام محل قرار گیرد؟ جواب دادن به سؤال هائی از این قبیل به بحث الگوریتم های انتخاب بستگی دارد.

روش های مدیریت فضای آزاد

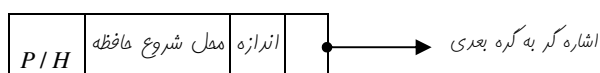
الف. روش نگاشت بیتی (Bit maps)

در این روش یک واحد مناسب در نظر گرفته می شود، و متناظر با این واحد یک بیت در نظر گرفته می شود، بنابراین با شروع از ابتدای حافظه هر واحدی از حافظه که پر باشد بیت متناظر آن 1 و اگر خالی باشد بیت متناظر آن صفر در نظر گرفته می شود. مشکل این روش این است که اگر پردازه به k واحد نیاز داشته باشد، می بایست در رشته بیتی حافظه به دنبال k صفر کنار هم باشیم. به عنوان مثال اگر واحد انتخابی $1k$ باشد. شکل زیر این موضوع را روشن می کند.



ب. روش لیست پیوندی (Linked list)

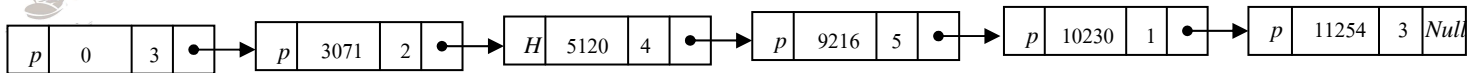
در روش لیست پیوندی با شروع از ابتدای حافظه به ازای هر پردازه یا فضای خالی یک گره ای با ساختار زیر در نظر گرفته می شود.



P : پردازه

H : فضای آزاد

مثال قبلی به روش لیست پیوندی



الگوریتم های انتخاب ها:

1- اولین مناسب (frist fit)

وقتی پردازش 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 تفصیص از آن محل صورت گرفته است. بدین ترتیب یکنواختی توزیع برنامه ها 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 مغره که به اندازه کافی بزرگ است به پردازش تفصیص دارد می شود. این روش باعث می شود که کوچکترین مغره بر اثر تفصیص باقی بماند. با این روش فضا های بزرگتر برای تقاضا های بیشتر حفظ می شوند. از آنجا که تمام لیست بلاک های آزاد باید بررسی شود، این تکنیک قدری زمانبر است.

پایان جلسه نهم

Pnu-Soal.ir