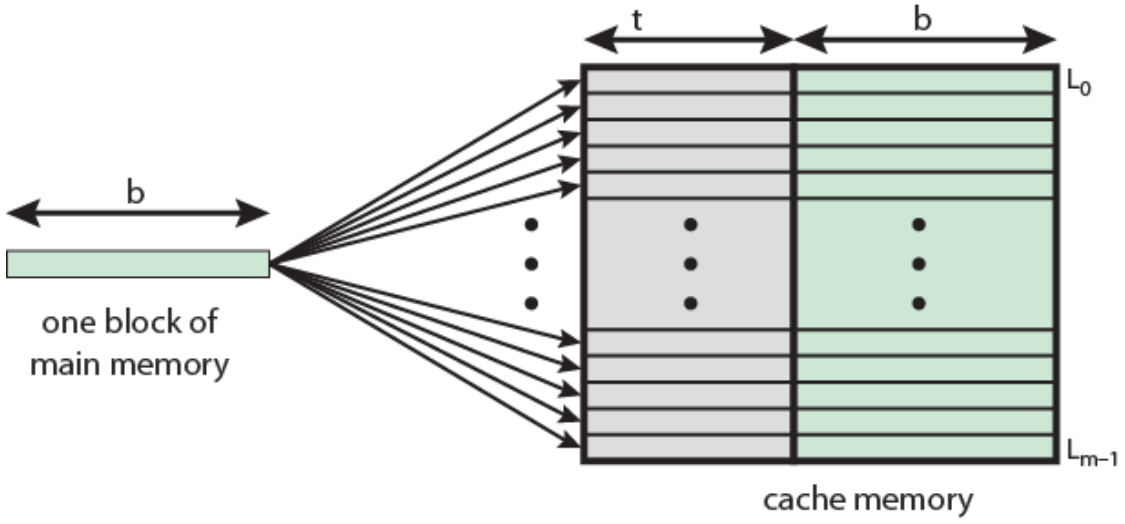
نقشه حافظه پنهان به روش انجمنی:
تناظر بین بلوک های حافظه اصلی و حافظه پنهان

- همانطور که گفته شد هر بلوک حافظه در حافظه اصلی می تواند در هر یک از خط های cache ذخیره شود.
- آدرس (شماره) بلوک در Tag ذخیره می شود.



مثال: نحوه تقسیم آدرس برای روش نقشه انجمنی

Tag 22 bit	Word 2 bit

- این مثال بر مبنای اطلاعات اسلاید ۷۳ معرفی شده است.
- ۲۴ بیت آدرس حافظه داریم. (چون ظرفیت حافظه اصلی ۱۶ مگابایت است.)
- ۲ بیت کم ارزش نشان دهنده آدرس کلمه در هر بلوک است. (هر بلوک ۴ بایت است.)
- ۲۲ بیت باقیمانده آدرس بلوک را نشان می دهد.
  - این ۲۲ بیت به عنوان Tag باید کنار هر خط ذخیره شود.
- نتیجه: اگر می خواهید وجود داده ای را در cache بررسی کنید:

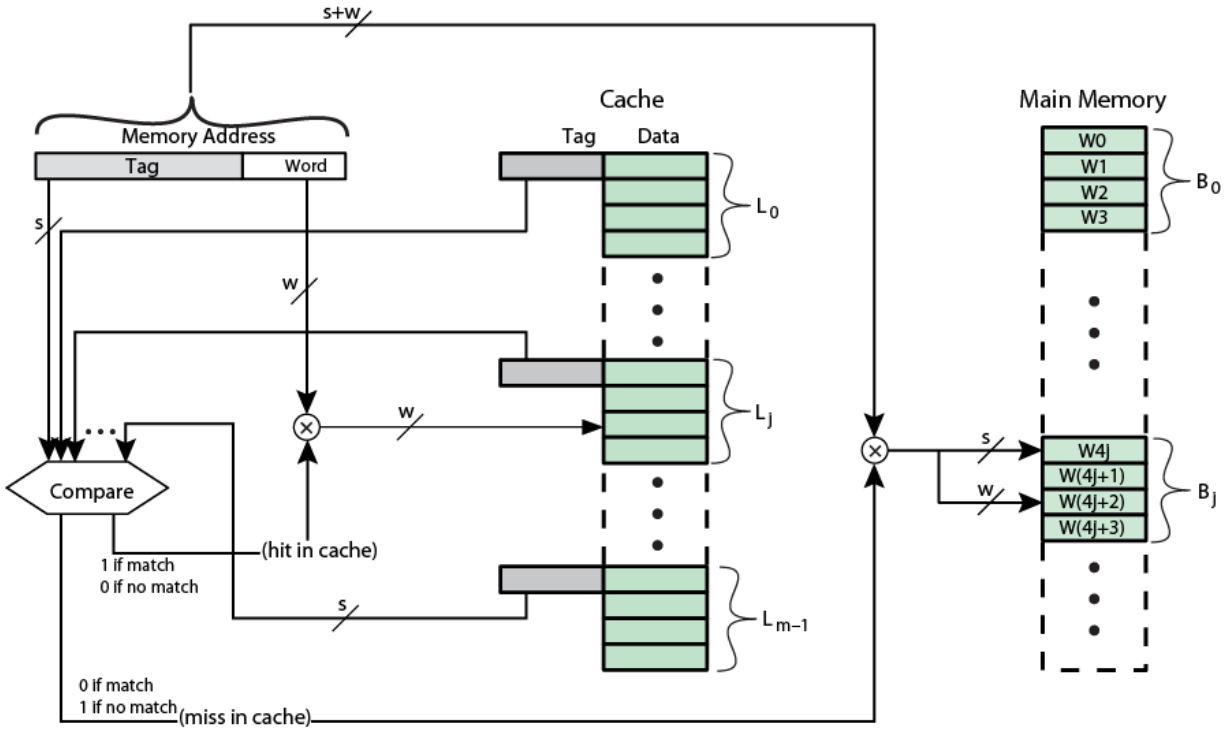
1. ابتدا s بیت آدرس آن را جدا کنید و با آن را با tag های همه خط های cache مقایسه کنید.
2. اگر tag ای برابر با این مقدار پیدا شد = داده در cache موجود است.
3. اگر tag ای برابر پیدا نشد = داده در cache موجود نیست و باید به سراغ حافظه اصلی برویم.

خلاصه روش نقشه انجمنی

- برتری و ضعف روش نقشه انجمنی
  - برتری: می توان داده را در هر مکان دلخواه حافظه پنهان ذخیره کرد. این باعث می شود که نیازی به جایگزینی داده هایی که شاید بسیار مفید هم باشند نباشد.
  - برتری: ظرفیت حافظه پنهان قابل افزایش است.
  - ضعف: مدار پیچیده خواهد بود زیرا باید تعداد زیادی Tag را با آدرس بلوک مقایسه کند.
  - ضعف: اندازه Tag هم بزرگ است. که بخش زیادی از حافظه پنهان را اشغال می کند.
- خلاصه مقادیر:
  - طول آدرس حافظه اصلی = s+w بیت.
  - تعداد کلمه های موجود در حافظه اصلی = 2^{s+w} کلمه.
  - اندازه بلوک = اندازه هز خط حافظه پنهان = 2^{w} کلمه.
  - تعداد بلوک ها در حافظه اصلی = 2^{s} بلوک.
  - تعداد خطها در حافظه پنهان = دلخواه
  - اندازه tag: s بیت.

مدار پیاده سازی شده نقشه انجمنی حافظه پنهان

این مدار الگوریتم پایین اسلاید ۸۲ را به شکل سخت افزاری انجام می دهد.



موارد اصلی در طراحی حافظه پنهان
۳- الگوریتم های جایگزینی

- اگر بخواهیم داده جدیدی را حافظه پنهان ذخیره کنیم کدام خط حافظه پنهان را باید پاک کنیم؟ الگوریتم جایگزینی این را توضیح می هد.
- در روش نقشه مستقیم:
  - هیچ انتخابی وجود ندارد. چون هر بلوک باید در خط اختصاصی خود ذخیره شود.
  - مجبور هستیم محتوای خط مربوطه را دور بریزیم.
- در روش نقشه انجمنی: انتخاب های زیادی وجود دارد و چندین روش موجود است.
  - خطی را که مدت زیادی است که خوانده نشده پاک کنیم. (روش LRU).
  - کهنه ترین خطی که نوشته شده را پاک کنیم. (روش FIFO).
  - خطی که کمتر از همه خوانده شده را پاک کنیم. (روش LFU).
  - به طور اتفاقی یک خط را پاک کنیم. (روش Random).

موارد اصلی در طراحی حافظه پنهان
۴- سیاست نوشتن

- اینکه حافظه پنهان کپی حافظه اصلی است بسیار می تواند دردسر ساز باشد.
- دردسر این است که الان دو نسخه از یک داده موجود است. یکی در **حافظه اصلی** و یکی در **حافظه پنهان**.

1. فرض کنید خطی را در حافظه پنهان تغییر دهید. این باعث می شود که نسخه اصلی این مقدار در حافظه اصلی دیگر معتبر نباشد.
2. ممکن است نسخه اصلی در حافظه اصلی تغییر کند. (دسترسی مستقیم دستگاه های I/O) بنابرین دیگر نسخه موجود در حافظه پنهان معتبر نیست.

- بنابرین باید نسخه موجود در دو طرف مرتب با یکدیگر به روز شوند.
- اینکه چگونه این کار را انجام دهند را می گویند: سیاست نوشتن. (write policy)

۴- سیاست نوشتن: دو روش عمده سیاست نوشتن وجود دارد.
WRITE TROUGH و WRITE BACK

Write Trough:

- هر بار که داده ای در حافظه پنهان نوشته می شود، همزمان در حافظه اصلی هم نوشته می شود.
- ارتباطات باس هم بررسی می شود. اگر مقدار خطی از داده های حافظه ی پنهان در حافظه اصلی تغییر کرد. حافظه پنهان هم به روز سانی می شود.
- این روش ترافیک زیادی روی باس ایجاد می کند و سرعت نوشتن را کند می کند.

Write Back

- داده هایی که در حافظه پنهان کپی شده اند، فقط در حافظه پنهان تغییر داده می شوند. کنار هر داده ای که تغییر کرده یک علامت گذاشته می شود.
- اگر قرار شد این خط پاک شود، آنگاه کل بلوک به حافظه اصلی کپی می شود.
- به دستگاه های دیگر هم می گوییم که داده هایی را که کپی شده اند را از حافظه پنهان بخوانند و به حافظه اصلی کاری نداشته باشند.
- در عمل در این روش بلوک کپی شده در حافظه اصلی را غیر معتبر می کنیم.
- کارایی کمی بهتر از روش اول دارد.

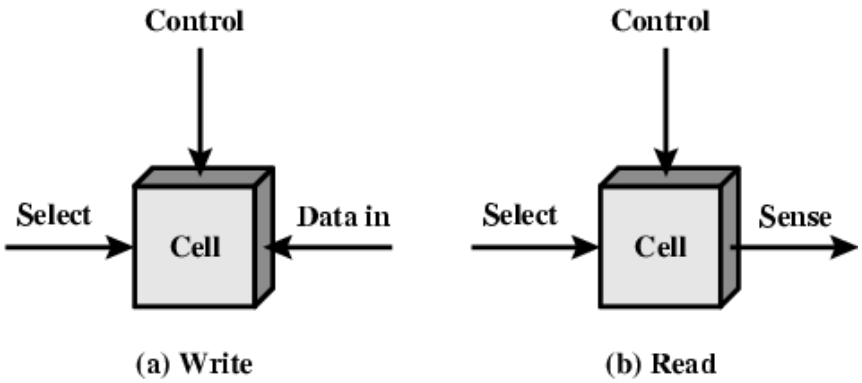
حافظه داخلی (اصلی) مرحله بعدی در سلسله مراتب حافظه کامپیوتر است.
انواع مختلف حافظه های نیمه هادی در جدول زیر آورده شده است.

- حافظه اصلی کامپیوتر ها امروزه اغلب با تکنولوژی نیمه هادی ساخته می شود.

نوع حافظه	دسته	روش پاک شدن	روش نوشتن	فرار بودن
Random-access memory (RAM)	حافظه نوشتنی و خواندنی	الکتریکی (هر بایت قابل پاک شدن است.)	الکتریکی	فرار (با قطع منبع تغذیه داده ها پاک می شوند)
Read-only memory (ROM)	حافظه فقط خواندنی	غیر قابل پاک شدن.	ماسک (اجزایی شبیه به فیوز سوزانده می شود.)	غیر فرار (داده ها بدون منبع تغذیه هم نگهداری می شوند.)
Programmable ROM (PROM)			الکتریکی	
Erasable PROM (EPROM)	حافظه اغلب خواندنی (حافظه قابل نوشتن هست ولی اغلب خوانده می شود.)	با نور UV، (کل حافظه باید پاک شود.)		
Electrically Erasable PROM (EEPROM)		الکتریکی (هر بایت قابل پاک شدن است.)		
Flash memory		الکتریکی (هر بلوک قابل پاک شدن است.)		

حافظه ها از سلول های حافظه تشکیل شده اند.
هر سلول حافظه باید دو عملیات را بتواند انجام دهد: خواندن، نوشتن

- هر سلول حافظه دارای چهار خط است: Select، Control، Data in، Sense.
- خط Select: سلول را فعال می کند.
- خط Control: تعیین می کند که این سلول را می خواهیم بخوانیم یا بنویسیم.
- خط Data in: برای نوشتن داده در سلول.
- خط Sense: برای خواندن از سلول حافظه.

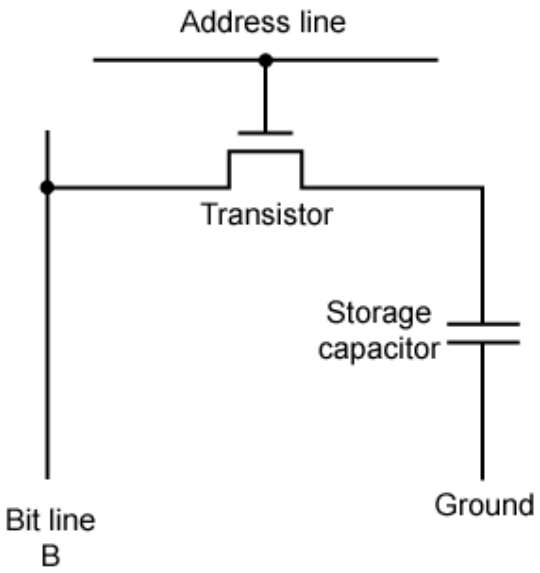


حافظه های RAM از دو نوع استاتیک و دینامیک ساخته می شوند.
حافظه دینامیک در سلول خود یک خازن به کار می برد.

- هر بیت با شارژ بودن یا دشارژ بودن خازن نشان داده می شود.
- شارژ خازن ها به مرور تخلیه می شود بنابرین نیاز به مدار به روز آوری وجود دارد.
- در مقایسه با نوع استاتیک ساخت هر بیت ساده تر است.
- اندازه فیزیکی هر بیت کوچکتر در می آید. بنابرین می توان ظرفیت بیشتری را در یک چیپ جا داد.
- هزینه هر بیت آن ارزانتر در می آید.
- نسبت به نوع استاتیک کندتر هستند. یعنی زمان دسترسی طولانی تری دارند.
- امروزه حافظه اصلی کامپیوترها همه از نوع RAM دینامیک هستند.

مدار یک سلول حافظه RAM دینامیک (مدار برای مطالعه)

- روی این شکل دو خط معمول سلول های حافظه یعنی Select (اینجا با عنوان آدرس) و Data in (اینجا با عنوان Bit line) دیده می شوند

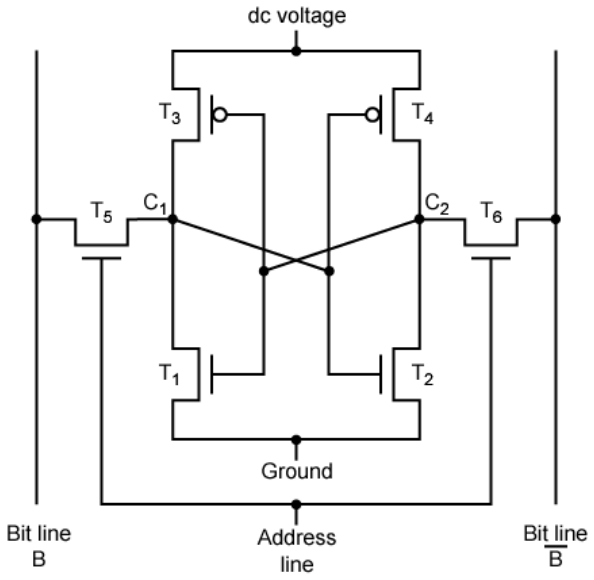


حافظه های RAM از دو نوع استاتیک و دینامیک ساخته می شوند.
حافظه استاتیک در سلول خود یک فلیپ فلاپ دارد.

- هر بیت با یک بودن یا صفر بودن خروجی فلیپ فلاپ ذخیره می شود.
- شارژی وجود ندارد که تخلیه شود بنابرین نیاز به مدار به روز آوری وجود ندارد.
- ساخت هر بیت آن نسبت به نوع دینامیک مشکل تر است.
- اندازه فیزیکی هر بیت بزرگتر در می آید. بنابرین می توان ظرفیت کمتری را در یک چیپ جا داد.
- هزینه هر بیت آن گرانتر در می آید.
- نسبت به نوع استاتیک سریعتر هستند. یعنی زمان دسترسی کوتاه تری دارند.
- امروزه حافظه Cache کامپیوترها همه از نوع RAM استاتیک هستند.

مدار یک سلول حافظه RAM استاتیک (مدار برای مطالعه)

- مشخص است برای ساخت این سلول تعداد ترانزیستور بیشتری به کار رفته است.



سازماندهی حافظه های نیمه هادی:
سازماندهی RAM دینامیک کمی متفاوت است.

- سازماندهی همه انواع مختلف حافظه های نیمه هادی به غیر از RAM دینامیک مانند آنچه است که تاکنون دیده ایم. (یادآوری از اسلاید ۶۰)
- یعنی: دو عامل مقدار ظرفیت را تعیین می کند. = $2^M xN$

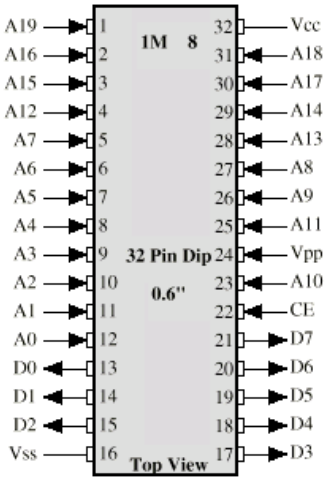
۱- تعداد مکان های حافظه = 2^M (M تعداد خط آدرس)

۲- اندازه هر مکان حافظه = N (N تعداد خط داده)

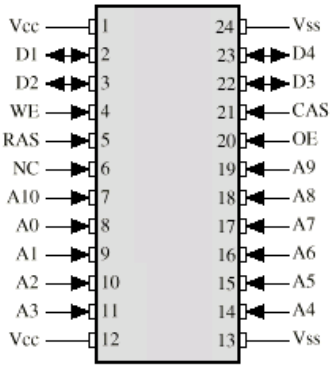
- در RAM دینامیک، خط های آدرس به دو بخش بالا و پایین تقسیم می شود.
- هر دو بخش روی یک مجموعه پایه فرستاده می شود. ولی در دو زمان متفاوت.
- به این ترتیب تعداد پایه های آدرس نصف می شود (M/2).
- دو پایه RAS و CAS روی چیپ حافظه قرار داده می شود.
- به کمک این دو پایه به حافظه می گوییم که درحال حاضر کدام بخش آدرس را روی پایه ها گذاشته ایم.

مقایسه پایه های یک DRAM (دینامیک) با یک حافظه نیمه هادی از نوع غیر DRAM (در اینجا یک EPROM است.)

- سازماندهی EPROM:
  - سازماندهی: 2^{20}x8
  - ۸ خط داده.
  - ۲۰ خط آدرس.
  - ظرفیت = 8 Mbit.



(a) 8 Mbit EPROM

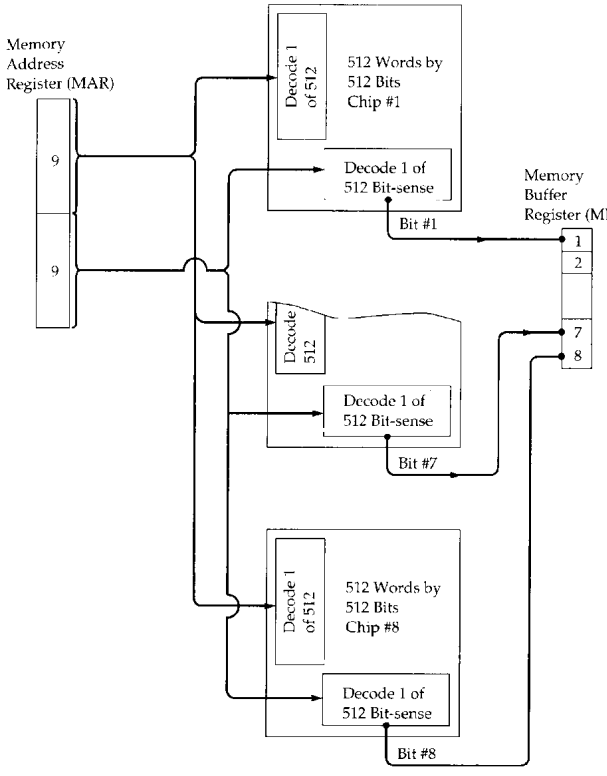


(b) 16 Mbit DRAM

- سازماندهی DRAM:
  - سازماندهی: 2^{22}x4
  - ۴ خط داده.
  - ۱۱ خط آدرس ظاهری.
  - ۲۲ خط آدرس واقعی.
  - ظرفیت = 16 Mbit.

ساختار داخلی یک حافظه نیمه هادی اعم از دینامیک و غیر دینامیک

- مدار داخلی حافظه ها به طور عمومی شکلی مانند روبرو است.
- سلول های حافظه به صورت آرایه ای مربعی کنار هم شده اند.
  - در اینجا ۸ تا آرایه ی 512x512 داریم.
- ابتدا آدرس در داخل به دو بخش تقسیم می شود.
- این دو بخش به دو دیکودر در سطر و ستون آرایه ها داده می شود.
- خروجی دیکودرها بیت مورد نظر در آرایه ها را انتخاب می کند.
- خروجی این ۸ آرایه ها کنارهم قرار می گیرد و بایت خروجی حافظه را تشکیل می دهد.
  - سازماندهی این حافظه = 2^{18}x8
  - ظرفیت این حافظه = 256Kx8 = 2Mbit



حافظه های خارجی کندترین نوع حافظه هستند
اما از نظر هزینه انجام شده برای هر بیت ارزانترین هستند.

- به دلیل سرعت پایین، CPU مستقیماً با حافظه های خارجی در ارتباط نیست.
- CPU برنامه ها را از حافظه اصلی اجرا می کند و داده ها نیز از همان جا برمی دارد.
- حافظه های خارجی تنها حکم انبار نگه داری داده ها و برنامه ها را دارند.

- انواع حافظه های خارجی:
  - دیسک مغناطیسی: هارد دیسک ها (داخلی و خارجی)، Floppy disc.
  - نوری: CD-ROM، CD-R/W، DVD-ROM، DVD، Blu-ray.
  - نوار مغناطیسی.

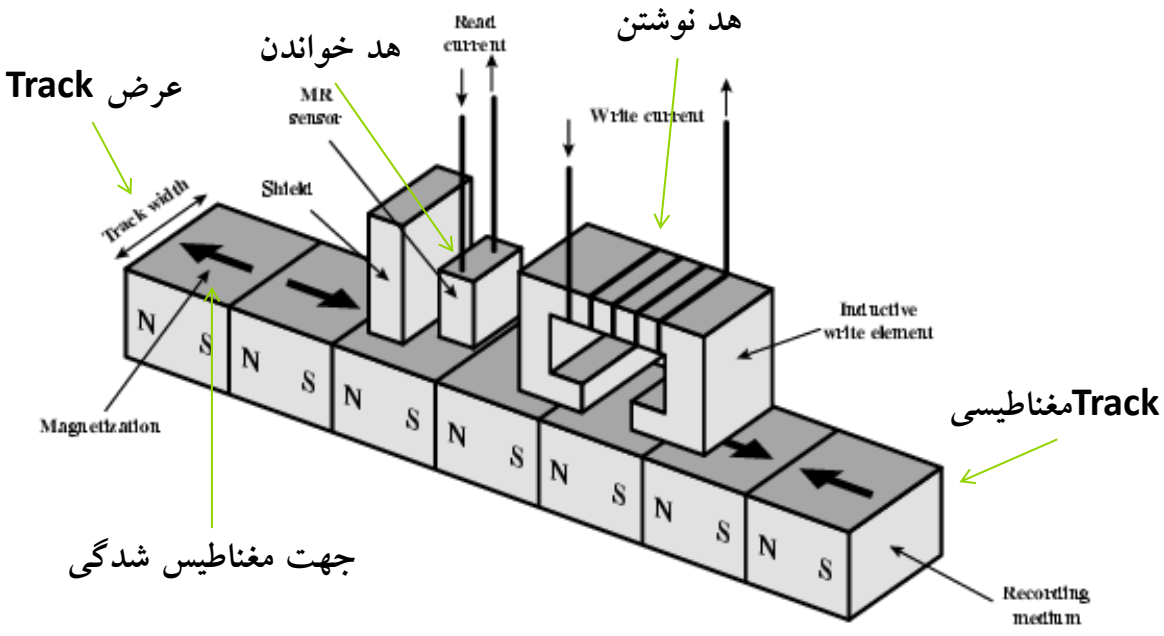
دیسک مغناطیسی

- دیسک های مغناطیسی یک حلقه (غیر مغناطیسی) هستند که با یک ماده مغناطیس شونده (مانند اکسید آهن) پوشانده می شوند.
- حلقه ابتدا آلومینیومی بود ولی امروزه اکثراً از شیشه ساخته می شود.
- نوشتن و خواندن با یک سیم پیچ القایی با نام head انجام می شود.
- ممکن است هد خواندن و نوشتن یکی باشد یا مجزا باشند.
- هنگام خواندن و نوشتن هد ثابت است و حلقه می چرخد.

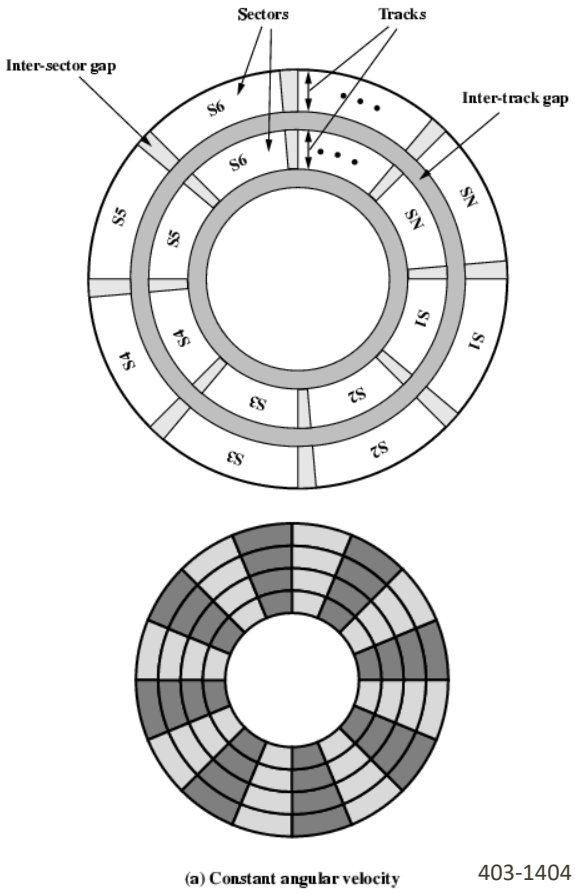
روش خواندن و نوشتن دیسک مغناطیسی

- نوشتن
  - سیم پیچ با ایجاد میدان مغناطیسی جهت آهنربایی هر بخش را تغییر می دهد.
  - هر بیت به طور مجزا در هر بخش حلقه قرار می گیرد.
- خواندن (روش سنتی)
  - عبور آهنربا های کوچک از زیر هد جریان الکتریکی در همان سیم پیچ نوشتن ایجاد می کند.
  - جهت جریان مقدار بیت را نمایش می دهد.
- خواندن (روش نوین)
  - هد مجزایی برای خواندن قرار داده شده است. در نزدیکی هد نوشتن.
  - در این هد از ماده ای استفاده شده که مقاومت الکتریکی آن وابسته به جهت میدان مغناطیسی است.
  - فرکانس کارکرد این هد بالاتر است. پس می توان به بیت های کوچکتر و در نتیجه ظرفیت بالاتر رسید.

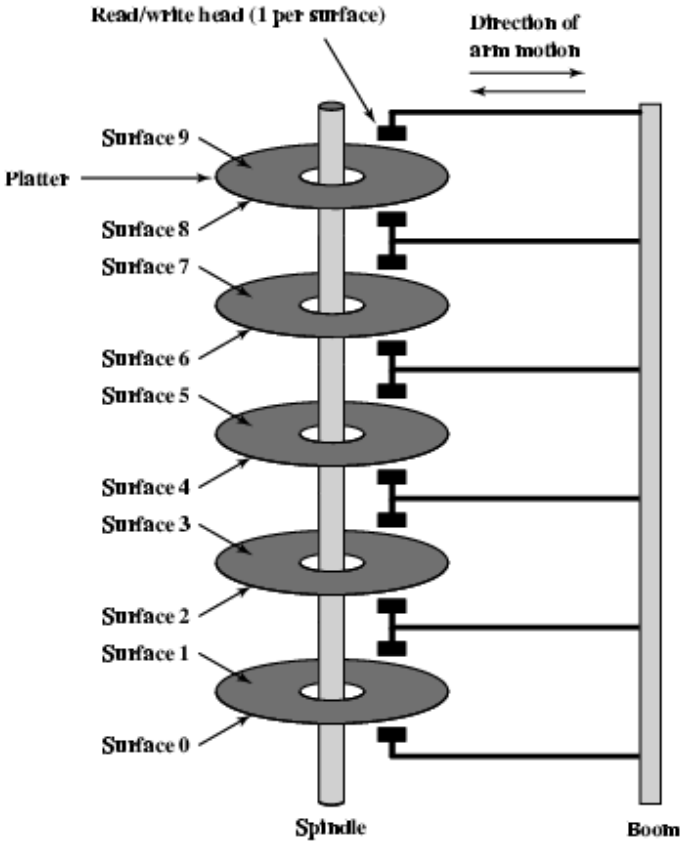
روش خواندن و نوشتن دیسک مغناطیسی (ادامه)



روش نگه داری داده ها روی دیسک مغناطیسی (روش اول)



- داده ها روی حلقه های به نام Track نگه داری می شوند.
- بین Track ها فاصله وجود دارد.
- با کاهش این فاصله ها می توان ظرفیت را افزایش داد. (هد باید دقیق تر شود.)
- تعداد بیت روی همه Track ها ثابت است.
- سرعت چرخش دیسک ثابت است.
- هر حلقه به چند sector تقسیم می شود. بین sector ها هم فاصله است.
- به هر sector شماره ای و الگوی مجزا تخصیص داده می شود تا قابل پیدا کردن باشند.
- بیت ها در حلقه های بیرونی از نظر فیزیکی طولانی تر اند.
- در روش دوم سرعت حرکت متغیر شده در track های بیرونی تعداد بیت های بیشتری ذخیره می شود.



- هارد دیسک های امروزی دارای چندین دیسک و چندین هد هستند که همزمان با هم کار می کنند.
- روی هر دیسک هم دو لایه مغناطیسی قرار داده شده است.
- هد ها معمولاً به هم متصل اند و مستقل نیستند.

معیارهای کارایی دیسک های مغناطیسی

- زمان Seek: (T_s)
  - زمان مورد نیاز برای رسیدن هد به track مورد نظر. میانگین این زمان برای سیستم های امروزی برابر کمتر از 10 ms است.
- تاخیر چرخش:
  - زمانی که مورد نیاز است تا هد روی بیت مورد نظر برسد. این زمان مرتبط با سرعت چرخش دیسک است $\frac{1}{2r}$
  - مثال: اگر سرعت چرخش 20000 rpm باشد، زمان یک چرخش کامل می شود: 3 ms و میانگین تاخیر چرخش می شود: 1.5 ms.
- پس اینکه هد به بیت مورد نظر رسید انتقال داده ها را به شکل پشت سرهم انجام می دهد. زمان لازم برای خواندن به b بیت اینگونه می شود:

$$T = \frac{b}{rN}$$

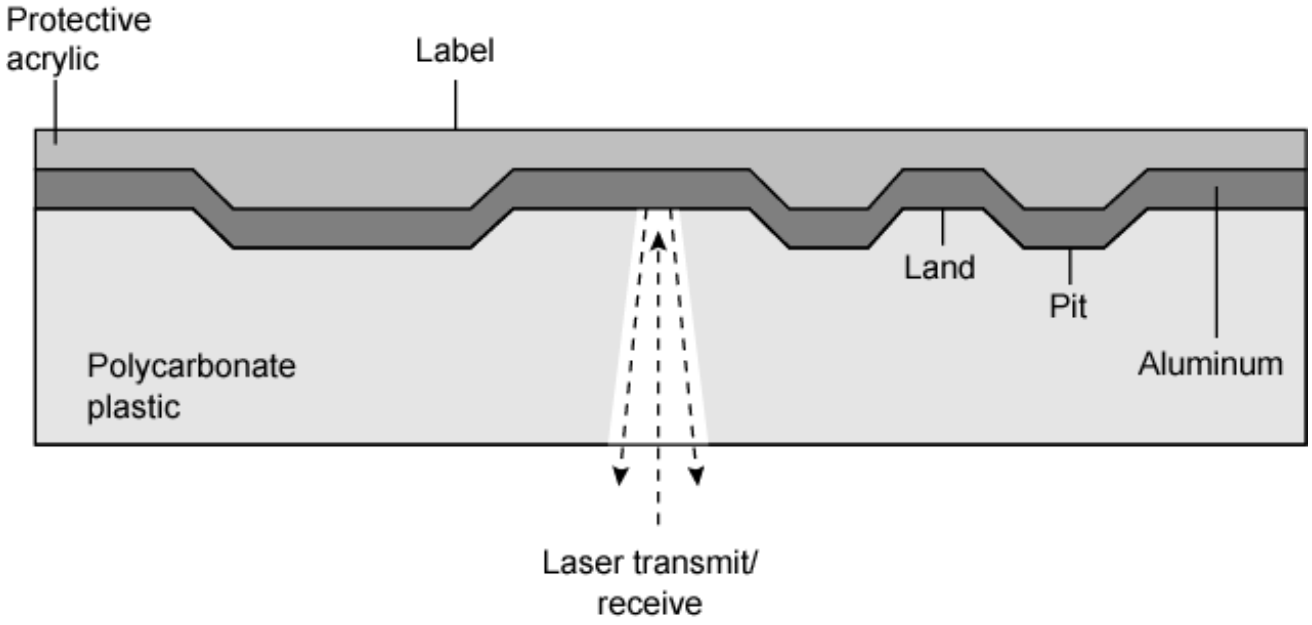
- r= سرعت چرخش (دور در ثانیه)، N = تعداد بیت در هر Track
- پس زمان دسترسی به b بیت = زمان Seek + تاخیر چرخش + زمان خواندن.

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

حافظه خارجی نوری

- حلقه پلی کربنات (پلاستیک شفاف) که یک لایه بازتاینده مثل آلومینیوم روی آن قرار گرفته برای نگهداری داده ها به کار می رود.
- داده ها به صورت چاله هایی روی لایه بازتابنده حک می شود.
- با کمک نور بازتابی از لیزر هر بیت خوانده می شود.
- هر بیت از نظر اندازه فیزیکی ثابت است.
- بنابرین حلقه های بیرونی ظرفیت بیشتری خواهند داشت.
- سرعت چرش دیسک باید متغیر باشد. (سرعت عبور هد از روی بیت ها ثابت است.)
  - اگر هد روی حلقه های بیرونی باشد سرعت چرخش کاهش می یابد.
  - اگر هد روی حلقه داخلی باشد سرعت افزایش می یابد.
- ذخیره داده ها به شکل یک رشته پشت سرهم مارپیچی قرار داده شده است.

روش خواندن حافظه های نوری

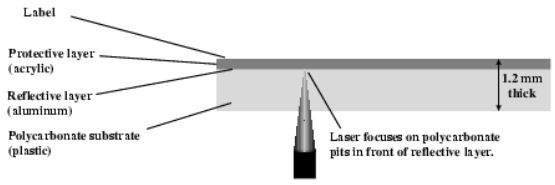


انواع مختلف دیسک های نوری

نام	ظرفیت	سرعت خطی پایه	طول فیزیکی رشته بیت ها	خواندنی یا نوشتنی
CD-ROM	650Mbyte	1.2 m/s (1x)	5.27 km	فقط خواندنی
CD-R	650Mbyte	1.2 m/s (1x)	5.27 km	قابل یکبار نوشتن
CD-RW	650Mbyte	1.2 m/s (1x)	5.27 km	قابل نوشتن مجدد
DVD	یک لایه 4.7 GB	1.2 m/s (1x)		قابل یکبار نوشتن
HD-DVD	یک لایه 15 GB	1.2 m/s (1x)		قابل یکبار نوشتن
Blu-ray	یک لایه 25 GB	1.2 m/s (1x)		قابل یکبار نوشتن

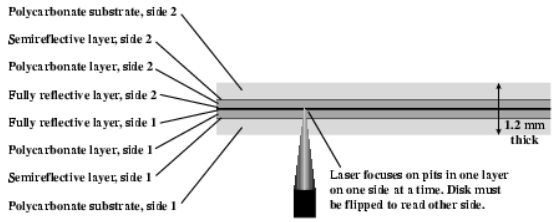
- در تکنولوژی های جدید تر فرکانس نور لیزر افزایش یافته است. (از رنگ قرمز به سمت آبی و UV رفته است.)
- زمان دسترسی: مشابه دیسک مغناطیسی است. + زمان لازم برای تنظیم سرعت چرخش

مقایسه CD و DVD و Blu-ray (در DVD داده به هد نزدیک تر شده است.)



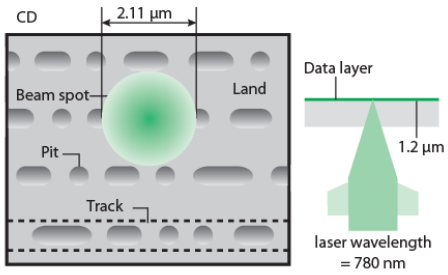
CD

(a) CD-ROM - Capacity 682 MB

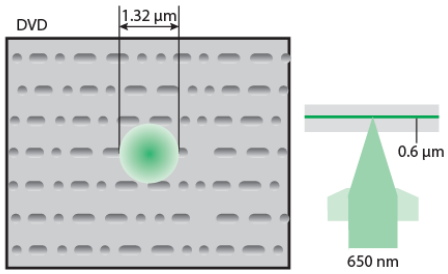


DVD

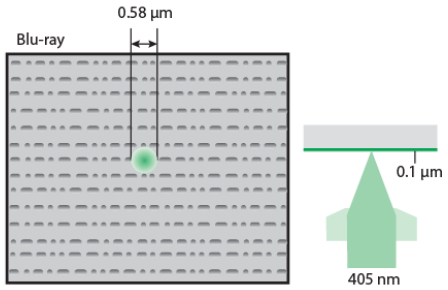
(b) DVD-ROM, double-sided, dual-layer - Capacity 17 GB



CD



DVD



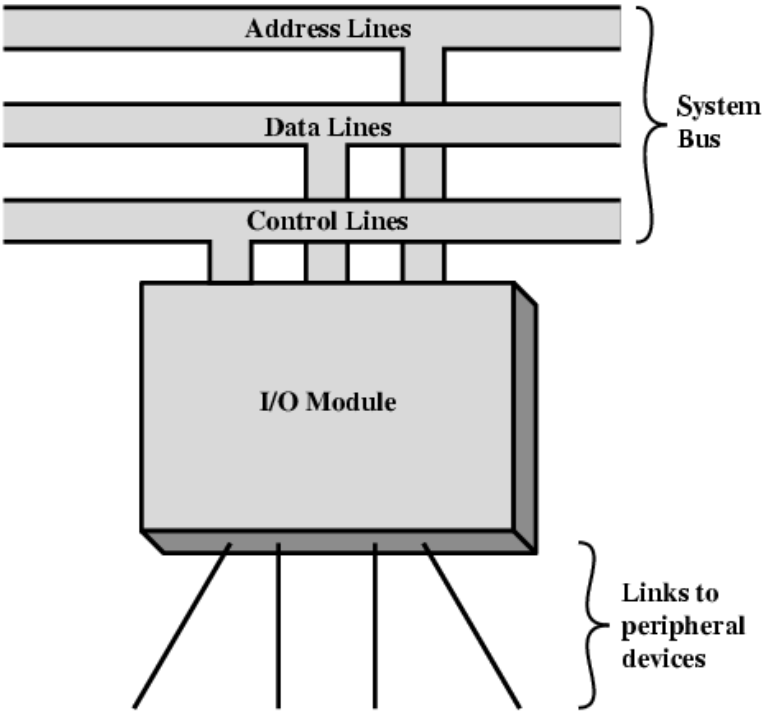
Blu-ray

- قطر تمرکز لیزر در تکنولوژی های جدید کاهش یافته بنابرین فاصله track ها و بیت ها کاهش یافته که این باعث افزایش ظرفیت شده است.

ماجول های I/O برای استاندارد سازی ارتباط میان CPU و حافظه با دستگاه های ورودی و خروجی معرفی شده اند

- دستگاه های جانبی (ورودی/خروجی) متعدد و گوناگونی باید به کامپیوتر وصل شوند.
  - نمونه ها: صفحه کلید، صفحه نمایش، ماوس، مودم، پورت های مختلف، کارت شبکه، ....
  - هر کدام حجم متفاوتی از داده را تولید می کنند یا دریافت می کنند. (یا هر دو)
  - هر کدام سرعت های تولید یا دریافت داده مختلفی دارند.
  - شکل تولید و دریافت داده هر کدام متفاوت است.
- ولی همه از CPU و RAM کندتر هستند. (زمان CPU برای صحبت مستقیم با اینها تلف خواهد شد.)
- برای صحبت با دستگاه های مختلف یک واسط یا منشی نیاز است.
- به این واسط می گوییم ماجول I/O.
- ماجول های I/O باید با دو طرف در ارتباط باشند.
- از یک سو با دستگاه های جانبی. از سوی دیگر با CPU یا حافظه ها.

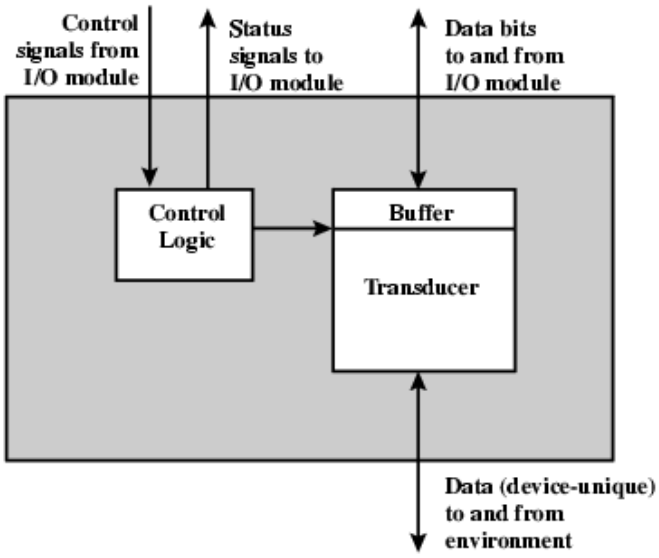
شکل عمومی اتصال ماجول های I/O به باس سیستم و دستگاه های جانبی



دستگاه های ورودی/خروجی (I/O)

- ارتباط با انسان
  - صفحه نمایش، صفحه کلید، پرینتر، ...
- ارتباط با سایر ماشین ها
  - کنترل و مونیتورینگ سیستم ها، مانند ارتباط با سنسورها و عملگرها در محیط صنعتی.
- ارتباط با سایر کامپیوترها (مخابرات)
  - مودم، کارت شبکه، کارت شبکه بی سیم، ....
- موضوع درس ما بررسی این دستگاه ها نیست (به دلیل گستردگی)
- ما در این بخش با ماجول های I/O آشنا خواهیم شد.

بلوک دیاگرام عمومی دستگاه های I/O (که بیش از این آنها را بررسی نخواهیم کرد.)



- Transducer: عنوان وسیله ای است که با جهان بیرون در ارتباط است و داده ها را در یک بافر (محل موقت) ذخیره می کند.
  - مثال: میکروفون و تقویت کننده آن در کارت صوتی.
- داده ها به همراه وضعیت ها (status) برای ماجول های I/O فرستاده می شود.
- ماجول های I/O هم فرمان های کنترلی را برای این دستگاه ها می فرستند.
- وضعیت: شامل وضعیت فعلی بافر داده است.

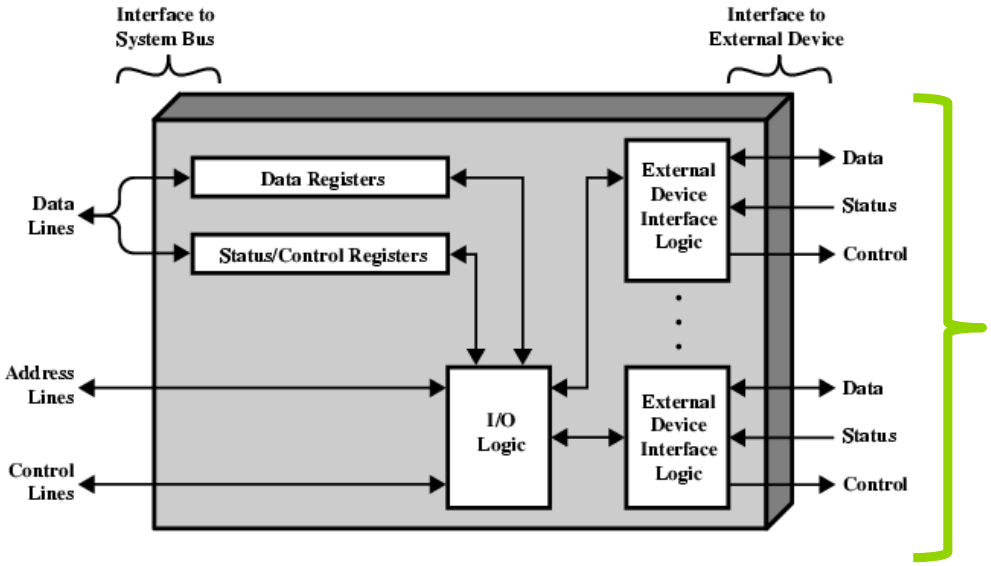
ماجول I/O چه کارهایی باید انجام دهد؟

- کنترل و زمانبندی
  - برای ایجاد هماهنگی میان دستگاه های خارجی و بخش های داخلی (به دلیل سرعت های متفاوت). که این شامل صحبت با هر دو طرف و هماهنگی با هر دو طرف است.
- ارتباط با CPU
  - CPU برای ماجول های I/O فرمان می فرستد، داده می فرستد و می گیرد. که تا حدود زیادی شبیه حافظه ها است (هر دستگاه دارای آدرس است.). با این تفاوت که وضعیت دستگاه ها برای CPU فرستاده می شود (وضعیت شامل مشغول بودن، پر و خالی بودن بافر و ...).
- ارتباط با دستگاه های خارجی که در اسلاید قبل توضیح داده شد.
- بافر کردن (نگهداری موقتی) داده ها
  - به دلیل تفاوت سرعت، ماجول فضایی برای نگهداری موقتی داده ها دارد. دستگاه ها توان دریافت سریع داده از CPU را ندارند، از سوی دیگر CPU برای دریافت مستقیم داده ها معطل خواهد شد چون طرف مقابل کند است.
- تشخیص خطا: ماجول I/O باید خطا های دستگاه ها را به CPU گزارش کند.

بلوک دیاگرام کلی ماجول I/O

ارتباط با باس سیستم

ارتباط با دستگاه های خارجی



این ماجول از چند دستگاه خارجی پشتیبانی می کند.

روش های راه اندازی دستگاه های I/O

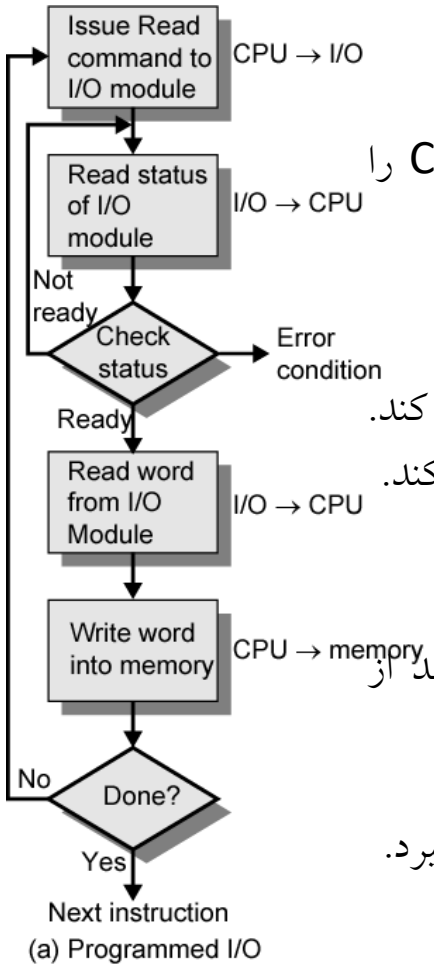
- بررسی و رسیدگی به دستگاه I/O به سه روش انجام می شود.

- Programed I/O (رسیدگی به I/O با برنامه)
  - رسیدگی به ماجول های I/O با کمک برنامه نوشته شده در CPU انجام می شود.
  - همه وظیفه بررسی وضعیت و انتقال داده از ماجول ها بر عهده CPU است.
  - CPU منتظر می شود تا عملیات ماجول I/O پایان یابد. (اتلاف زمان CPU)

- Interrupted I/O (رسیدگی به I/O با وقفه)

- DMA (Direct Memory Access) (دسترسی مستقیم به حافظه)

روند اجرای PROGRAMED I/O



- روند مقابل مراحل خواندن داده ای از I/O توسط CPU را نشان می دهد. روند نوشتن هم مشابه است.

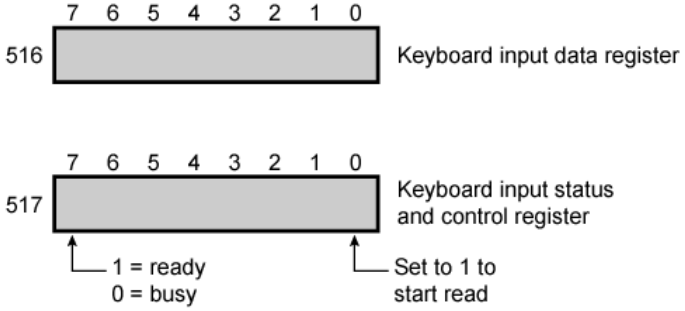
1. درخواست CPU از ماجول (با دستور در برنامه)
2. ماجول فرمان را اجرا می کند.
3. ماجول وضعیت (status) اجرای کار را مقداردهی می کند.
4. برنامه CPU در یک حلقه وضعیت را مرتباً بررسی می کند.
5. ماجول به تنهایی برای CPU وضعیت را نمی فرستد.
6. ماجول به CPU درخواستی برای وقفه نمی فرستد.
7. CPU می تواند برای وضعیت مطلوب صبر کند یا بعد از انجام چند دستور برای بررسی مجدد بازگردد.
8. پس از مطلوب بودن وضعیت داده ها خوانده می شود.
9. همه اینها در یک برنامه که کاربر می نویسد انجام می گیرد.

روند اجرای PROGRAMED I/O (ادامه)

- هر دستگاه دارای یک شماره (identifier) (آدرس) منحصر به فرد است.
- هر ماجول ممکن است از چندین دستگاه (چندین آدرس) پشتیبانی کند.
- از نظر خواندن و نوشتن و از نگاه CPU ماجول I/O شبیه یک حافظه است.
- دو روش معمول در آدرس دهی دستگاه های I/O وجود دارد.
  - Memory mapped: دستگاه ها و حافظه ها در یک فضای آدرسند. (پشت سر هم اند.) در این روش عملاً تفاوتی میان I/O و حافظه برای CPU نیست. (دستورهای مشترک دارند.)
  - Isolated: دو فضای مجزا برای حافظه و I/O داریم. دو آدرس مجزا داریم. دستور های مخصوص برای دسترسی به I/O قرار داده شده است.
- CPU فرمان های دیگری (به غیر از خواندن و نوشتن) نیز به ماجول های I/O می فرستد.
  - فرمان های کنترلی: اجرای انجام فرمان های مختلف در دستگاه I/O.
  - فرمان هایی برای بررسی وضعیت و بررسی خطاها.

مثال از برنامه ای برای ارتباط با صفحه کلید به روش PROGRAMED I/O

- ماجول I/O دارای دو مکان حافظه در آدرس های 516 و 517 است.
- در خانه 516 کلید فشار داده شده ذخیره شده است.
- در خانه 517 دو مقدار وجود دارد.
  - بیت هفتم نشان می دهد که تبدیل صفحه کلید انجام شده است یا نه ؟
  - بیت صفرم: بیت فرمان است. با نوشتن ۱ روی این بیت کار صفحه کلید برای خواندن کلید ها آغاز می شود.



مثال از برنامه ای برای ارتباط با صفحه کلید به روش PROGRAMED I/O (ادامه)

- Isolated

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

1. فرمان خواندن.
2. بررسی پایان کار.
3. اگر پایان نیافته، به خط قبلی برو.
4. خواندن داده ذخیره شده.

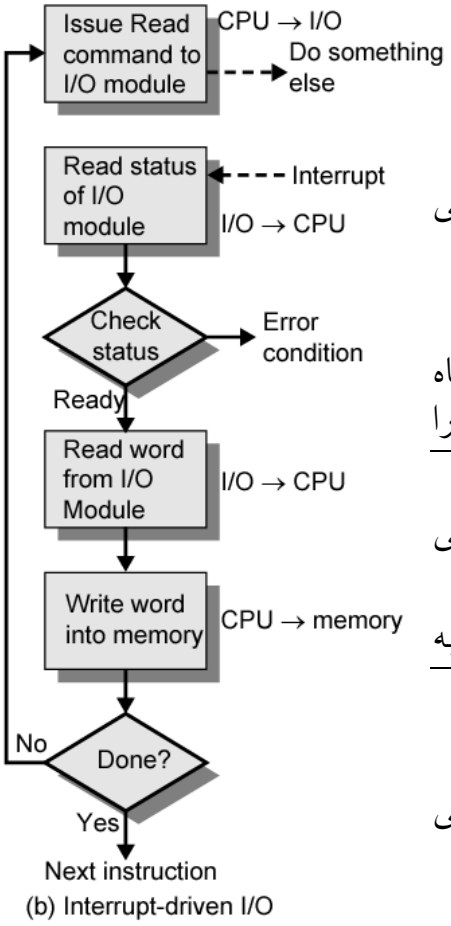
- memory mapped.

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

1. فرمان خواندن «۱» در AC ذخیره می شود.
2. 1 به 517 فرستاده می شود.
3. 517 خوانده می شود.
4. اگر آماده نیست به خط قبل برو.
5. مقدار 516 را بخوان.

روش دوم رسیدگی به ماجول I/O:
INTERRUPTED I/O (رسیدگی با پاسخ به درخواست وقفه)



(b) Interrupt-driven I/O

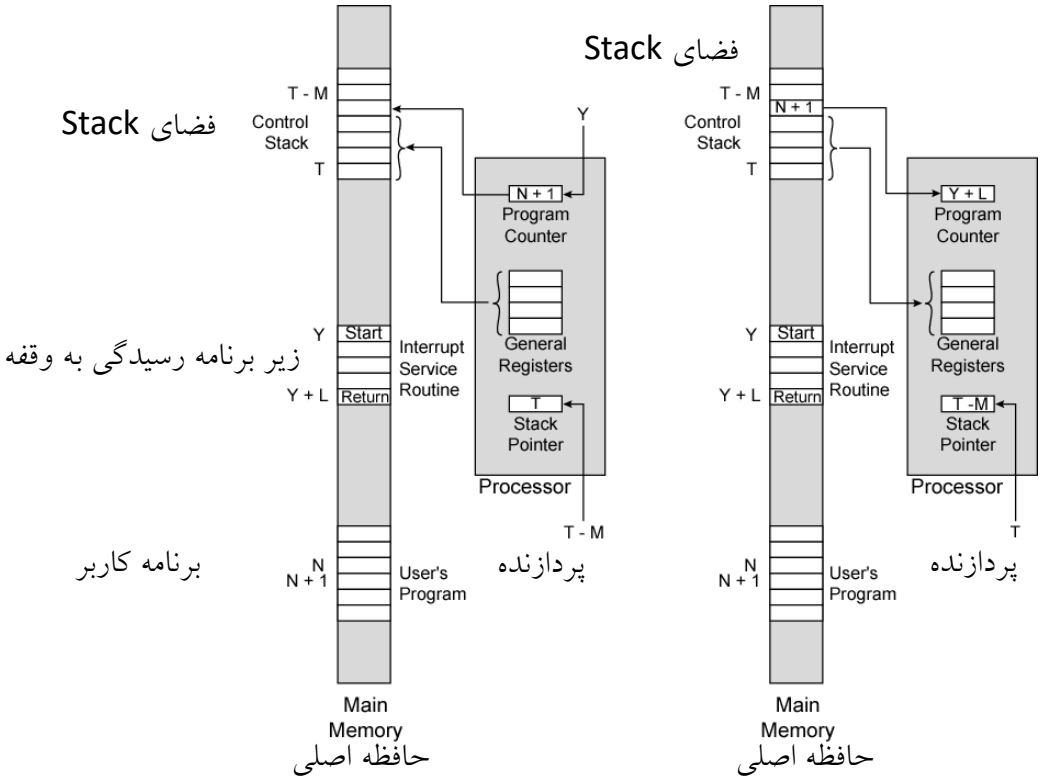
- رسیدگی به وقفه ها با پاسخ به درخواست وقفه انجام می شود.

1. CPU فرمان خواندن می دهد. (با اجرای دستور در برنامه)
2. ماجول I/O کار را شروع می کند (صحبت با دستگاه خارجی) در حالی که CPU می تواند اجرای برنامه خود را ادامه دهد.
3. پس ازآمادگی ماجول I/O از CPU درخواست وقفه می کند.
4. CPU با پاسخ به درخواست وارد زیر برنامه ی رسیدگی به ماجول می شود.
5. ماجول I/O داده ها را برای CPU می فرستد.
6. در پایان اجرای زیر برنامه CPU اجرای برنامه را از سرمی گیرد.

روش دوم رسیدگی به ماجول I/O:
INTERRUPTED I/O (ادامه)

- در این روش دیگر CPU انتظار پاسخ دستگاه را نمی کشد.
- بنابرین در برنامه نیز حلقه ای برای بررسی وجود ندارد.
- تنها CPU در انتهای اجرای هر دستور وقفه ها را بررسی می کند. (اسلاید ۳۹ و ۴۲)
- این کار به شکل سخت افزاری و بدون دخالت برنامه نویس انجام می شود.
- اگر وقفه پذیرفته شود:
  - مقدار PC ذخیره می شود. (سخت افزاری)
  - مقدار رجیسترها باید ذخیره شود. (نرم افزاری و با اجرای دستور)
  - معمولاً این ذخیره سازی در فضایی به نام Stack انجام می شود.
  - رسیدگی به ماجول انجام شود. (با انجام زیر برنامه)
  - دوباره رجیسترها و PC بازیابی می شوند.

تغییر مقدار حافظه و رجیسترها حین اجرای وقفه



بازگشت از وقفه

اجرای وقفه بعد ازاجرای دستوری در آدرس N

موارد مهم در روش INTERRUPT I/O

۱- CPUاز کجا باید بفهمد که کدام ماجول درخواست وقفه داده است؟

- چون چندین ماجول ممکن است به باس متصل باشد،CPU از کجا باید بفهمد که کدام ماجول درخواست وقفه داده است؟ وقتی درخواست وقفه می آید CPU می فهمد که درخواست وقفه آمده ولی نمی داند کدام ماجول این درخواست را فرستاده. پس باید به روشی مطلع شود.
- روش های اطلاع از آدرس (شماره) درخواست کننده:

1. اتصال سیم مجزا از هر ماجول به CPU
   - این روش ساده است. اما غیر عملی است. به دلیل محدود بودن پایه ها روی CPU.
2. بررسی نرم افزاری (software poll)
   - با برنامه ای از همه ماجول ها سوال کنیم: آیا شما بودی که درخواست وقفه دادی؟
   - این روش زمان گیری خواهد بود. (دقت کنید که این روش Programed I/O نیست.)
3. بررسی سخت افزاری (hardware poll) یا (daisy chain)
   - پس از دریافت وقفه CPU پایه ای را به نام INTA فعال می کند. (یعنی من وقفه را دریافت کردم.)
   - این پایه به همه ماجول ها متصل است. هر ماجولی که در خواست وقفه کرده باشد با دیدن این پیغام شماره (آدرس) خود را روی باس داده می فرستد.
4. روش داوری باس (bus Arbitration) (اسلاید ۵۶)

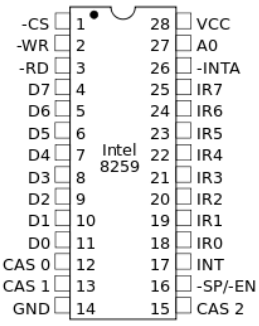
هر ماجول پیش از درخواست وقفه باید کنترل باس را دردست گیرد. (مثال: باس PCI)

موارد مهم در روش INTERRUPT I/O
۲- اگر چندین ماجول درخواست وقفه کنند CPU چه کار باید انجام دهد؟

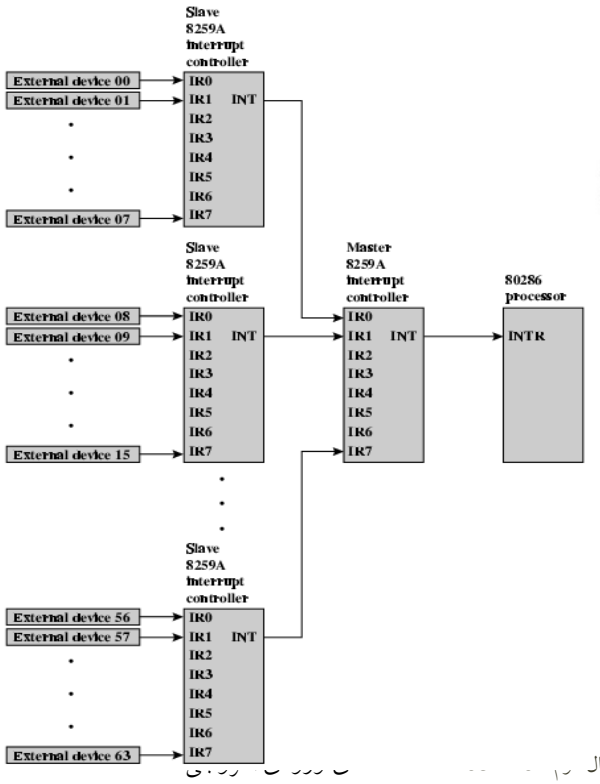
- این مطلب در اسلاید ۴۴ مطرح شد.
- یا وقفه های دیگر غیرفعال می شوند
- یا اولویت وقفه تعریف می شود.
- اولویت بالاتر می تواند در اجرای رسیدگی به اولویت پایین وقفه ایجاد کند.
- البته اگر روش داوری باس به کار رفته باشد دیگر مبحث اولویت نیست. هر ماجولی که باس را در اختیار داشته باشد می تواند درخواست وقفه بدهد.

نمونه از روش HARDWARE POLL
پردازنده خانواده x86 (8086) و کنترل کننده وقفه 8259A

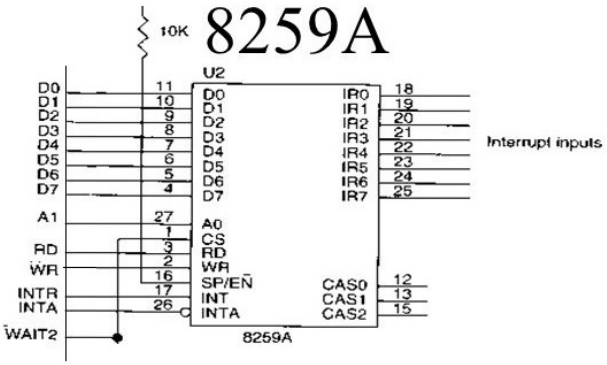
- برای پردازنده 8086 تنها یک پایه ورودی INT برای دریافت درخواست وقفه قرار داده شده است.
- برای دریافت چندین وقفه یک IC با نام 8259A معرفی شده است.
- این IC معروف به interrupt controller است و ۸ پایه ورودی برای وقفه دارد.
- یک پایه خروجی آن هم به INT در 8086 وصل می شود.
- نحوه کارکرد:
  - 8259A درخواست های وقفه را دریافت می کند.(تا ۸ درخواست.)
  - 8259A اولویت را بررسی می کند.
  - 8259A از طریق پایه INT به CPU خبر می دهد.
  - اگر CPU درخواست را قبول کند با پایه INTA تایید آن را به 8259A خبر می دهد.
  - حالا 8259A شماره وقفه مورد نظر را روی باس داده می گذارد تا CPU آن را بردارد.
  - CPU هم با توجه به شماره ای که دریافت کرده زیربرنامه رسیدگی مناسب را اجرا می کند.



- نحوه افزایش تعداد وقفه ها با روش Master/Slave



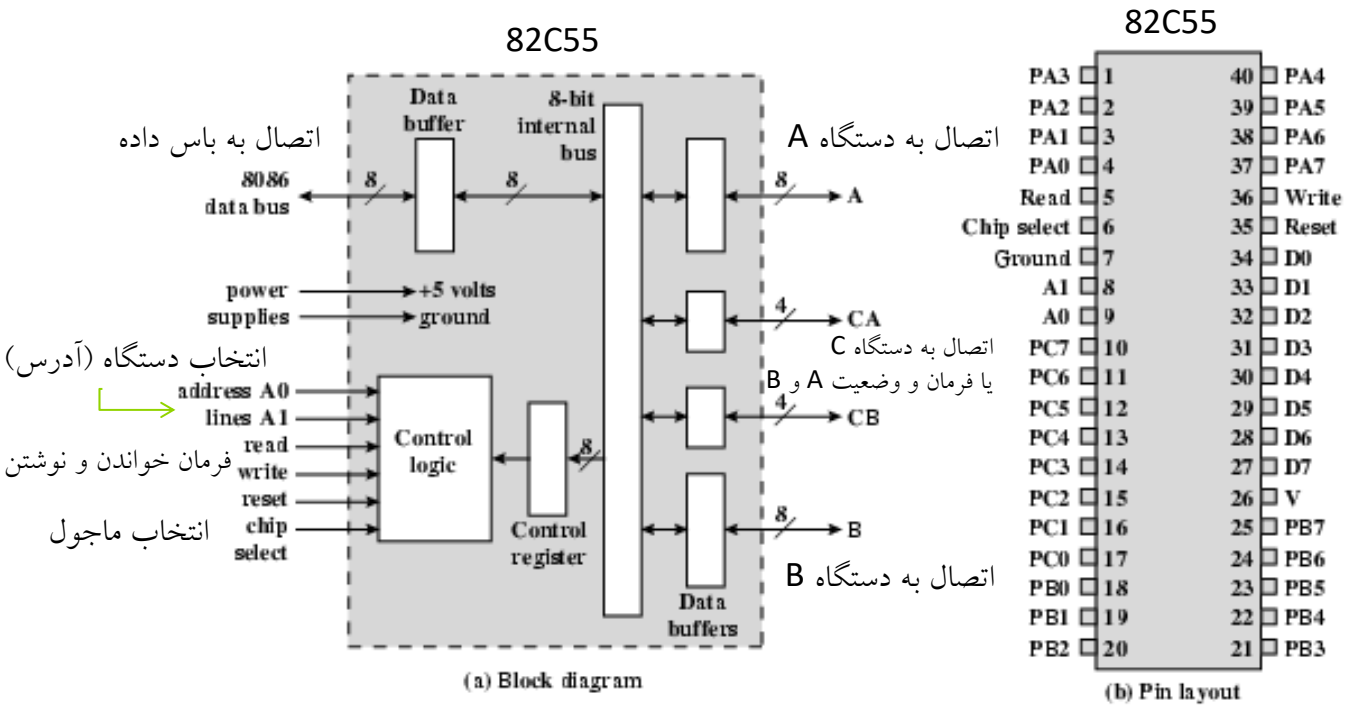
نحوه اتصال 8259A به 8086

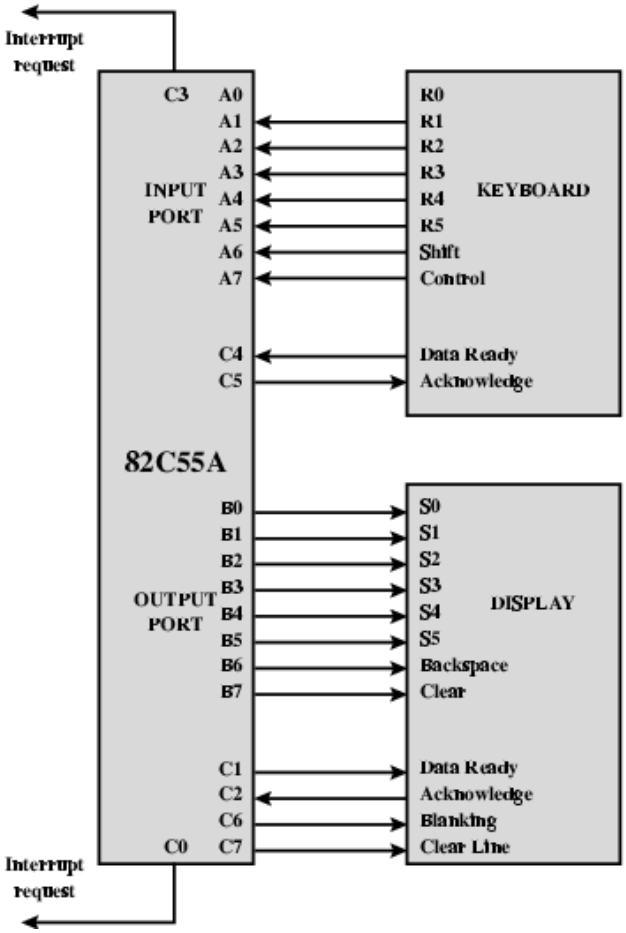


مثالی از یک ماجول I/O ساده: 82C55A
معروف به PPI (PROGRAMMABLE PERIPHERAL INTERFACE)

- در اینجا با یک ماجول I/O ساده و متداول آشنا می شویم.
- این ماجول قادر است با سه دستگاه ۸ بیتی در ارتباط باشد (دریافت و ارسال داده.)
- می توان صفحه کلید یا صفحه نمایش و ... را به این ماجول متصل کرد.
- هر دستگاه می تواند داده های ۸ بیتی برای ماجول بفرستد.
- ماجول داده های این دستگاه ها را ذخیره می کند و می تواند برای CPU بفرستد.
- با هر دو روش Programed I/O و Interrupt I/O می توان با آن کار کرد.
- همچنین می توان این ماجول را به گونه ای تنظیم کرد که با دو دستگاه ۸ بیتی در ارتباط باشد.
- در این حالت ۸ پایه باقیمانده می تواند برای ارسال فرمان ها به دو دستگاه و همچنین دریافت وضعیت آنها به کار رود.

پایه ها و بلوک دیاگرام 82C55 PPI





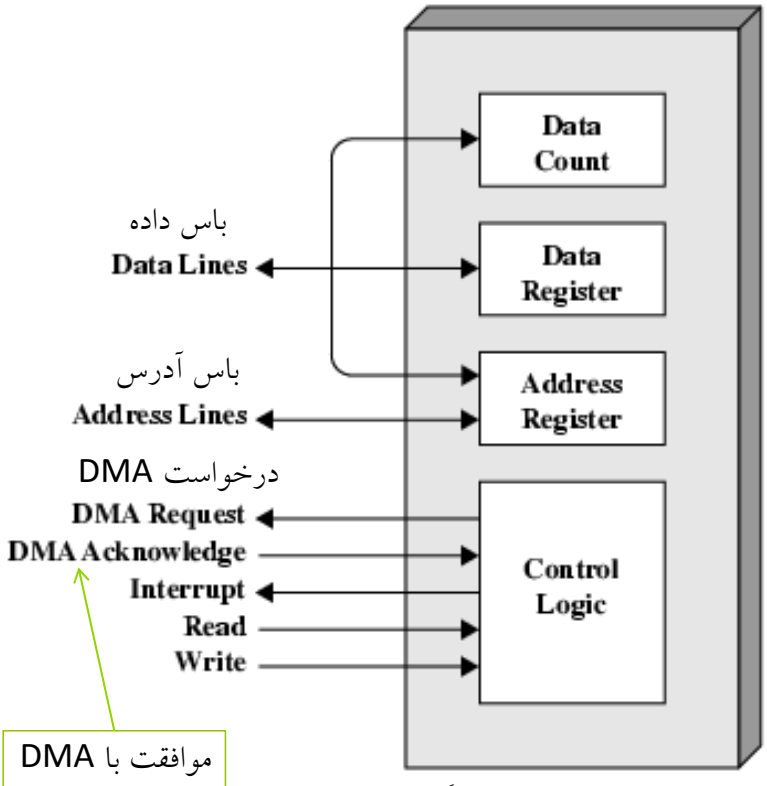
نمونه ای از اتصال 82C55 به دو دستگاه (یکی ورودی و دیگری خروجی) و اتصال پایه های فرمان و وضعیت این دو دستگاه

- پایه های C0, C3: درخواست وقفه از CPU
- پایه C6: فرمان پاک کردن صفحه.
- پایه C4: آماده بودن داده صفحه کلید.
- پایه C5: تایید دریافت داده توسط ماجول.
- پایه C1: اعلام وجود داده به صفحه نمایش.
- پایه C2: تایید دریافت داده توسط صفحه نمایش

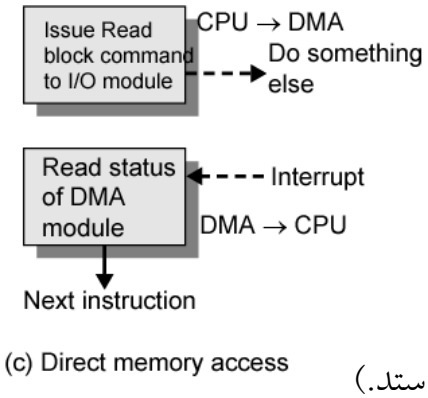
روش سوم رسیدگی به ماجول I/O:
DMA (Direct Memory Access)(دسترسی مستقیم به حافظه)

- هر دو روش قبلی زمان زیادی از CPU را می گیرند زیرا برای انتقال هر بایت (یا کلمه) باید CPU درگیر شود. (همه اطلاعات از داخل CPU می گذرد.)
  - نرخ انتقال اطلاعات هم محدود می شود.
- در خیلی از موارد حجم زیادی از داده باید از ماجول I/O به حافظه و یا برعکس فرستاده شود. (مثال: ضبط صدا، پخش ویدئو)
- در DMA به یک ماجول جدید که روی باس نصب شده اجازه می دهیم که باس را در اختیار بگیرد می دهیم تا انتقال اطلاعات بین حافظه و ماجول های I/O را انجام دهد.
- این ماجول جدید با نام ماجول DMA یا کنترل کننده DMA شناخته می شود.
- این ماجول معمولاً صبر می کند تا زمانی که CPU از باس استفاده نمی کند، باس را در اختیار بگیرد. به این ترتیب تاخیری در کار CPU ایجاد نمی شود.

نمای عمومی یک ماجول DMA



- این ماجول مانند همه بخش ها به باس متصل است. (باس داده و آدرس)
- فرمان های معمول به این ماجول وارد می شود. (خواندن و نوشتن)
- دو پایه درخواست از CPU دارد.
  - درخواست وقفه: مانند سایر ماجول های I/O
  - درخواست DMA: که درخواست در اختیار گرفتن باس است.
  - اگر CPU موافق باشد، اعلام موافقت را می فرستد.

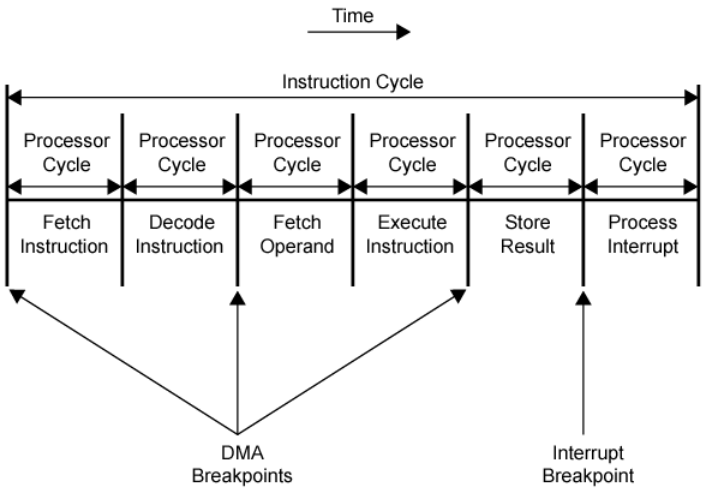


DMA چگونه کار می کند؟

- CPU به ماجول DMA می گوید:
  - بخوان یا بنویس (با توجه به کاربرد)
  - از این آدرس بخوان (آدرس ماجول I/O مورد نظر را می فرستد.)
  - آدرس شروع داده ها در حافظه را هم می فرستد.
  - مقدار داده ای را هم که باید انتقال داده شود هم می فرستد.
- CPU به اجرای ادامه برنامه خود می پردازد و دیگر کاری به انتقال داده ندارد.
- DMA انتقال را آغاز می کند. (از حافظه به ماجول I/O یا برعکس)
  - DMA این کار را زمانی انجام می دهد که باس بی کار باشد.
- هر وقت انتقال به طور کامل تمام شد. ماجول DMA برای CPU درخواست وقفه می فرستد.

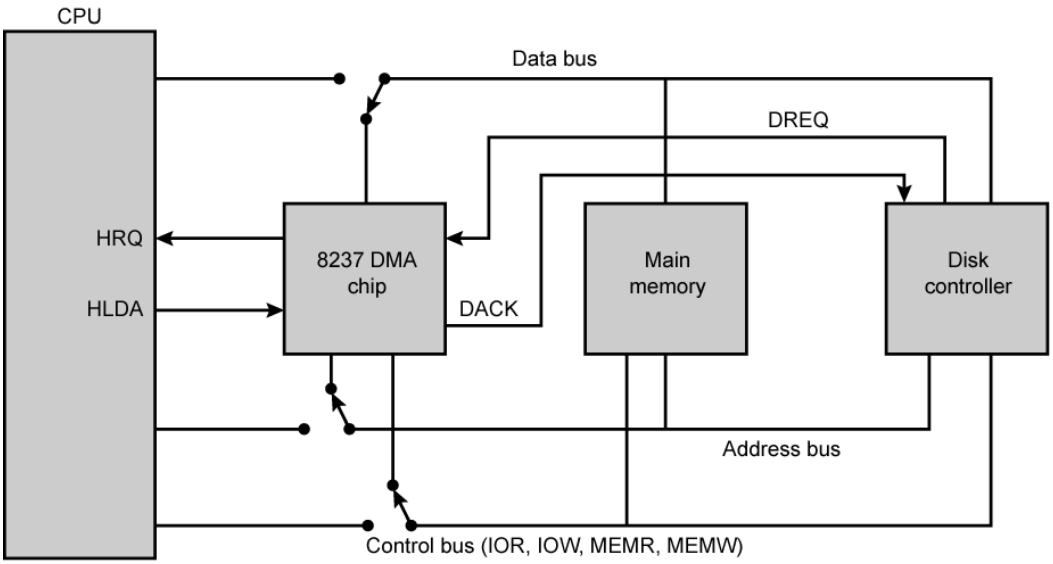
مثالی از روند اجرای کار DMA در حین کار عادی CPU (اجرای برنامه عادی)

- DMA در میان اجرای دستورهای CPU هر زمان که CPU از باس استفاده نمی کند کار خود را انجام می دهد. (اصطلاحاً به این می گویند دزدی سیکل باس.)
- شکل زیر اجرای یک دستور CPU را نشان می دهد.
- DMA در بخش رمزگشایی دستور و اجرای دستور از باس استفاده کرده است.

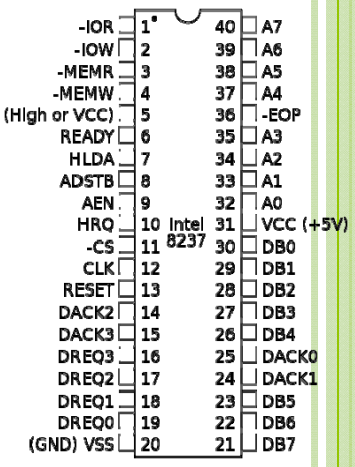


مثالی از یک کنترل کننده DMA: Intel 8237A
این ماجول برای پردازنده Intel 8086 طراحی شد.

نحوه اتصال این کنترل کننده DMA و CPU، یک ماجول I/O و حافظه



پایه های 8237



DACK = DMA acknowledge
DREQ = DMA request
HLDA = HOLD acknowledge
HRQ = HOLD request

مثالی از نحوه عملکرد Intel 8237A (انتقال داده بین یک ماجول I/O و حافظه بدون دخالت دادن CPU)

- مثال: می خواهیم حجم زیادی از داده را از حافظه برداریم و در ماجول I/O (که در اینجا با یک دیسک مغناطیسی در ارتباط است.) ذخیره کنیم.

1. ابتدا CPU درخواست انجام این کار برای ماجول I/O می فرستد.
2. این ماجول با فعال کردن پایه DREQ از 8237 درخواست عملیات DMA می کند.
3. 8237 با فعال کردن پایه HRQ از CPU درخواست باس می کند.
4. CPU پس از اجرای سیکل فعلی دستور، پایه HDLA را فعال می کند. یعنی با درخواست موافقت شده و باس آزاد شده است.
5. 8237 با فعال سازی پایه DACK به ماجول I/O می گوید که برای انجام عملیات آماده است.
6. 8237 انتقال اطلاعات را آغاز می کند. اولین آدرس حافظه را روی باس آدرس می گذارد و فرمان خواندن به حافظه و فرمان نوشتن به ماجول I/O می دهد. ماجول I/O داده را از روی باس داده بر می دارد.
7. در مرحله بعدی 8237 با افزایش یک مقداری آدرس حافظه همین کار را تکرار می کند تا عملیات پایان یابد. (مقدار داده ای که می خواستیم انتقال دهیم پایان یابد.)
8. در پایان کار، 8237 پایه HRQ را غیر فعال می کند تا CPU بفهمد و باس را پس بگیرد.

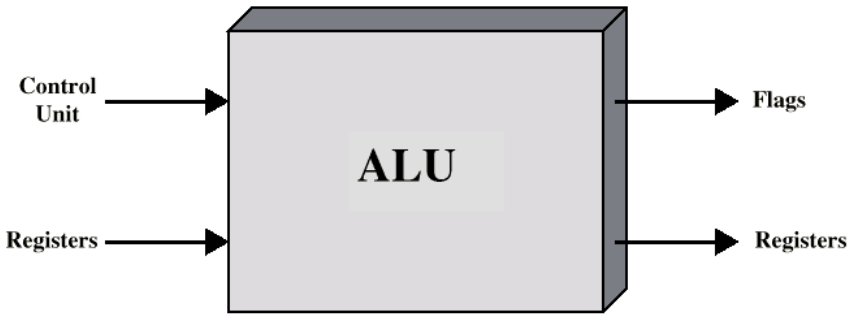
واحد پردازش مرکزی (CPU)

شامل: محاسبات کامپیوتر، مجموعه دستورها (خصوصیات و عملکرد)، مجموعه دستورها (مودهای آدرس دهی)، ساختار وکارکرد پردازنده، کامپیوترهای RISC

بخش سوم

واحد حسابی و منطقی (ARITHMETIC & LOGIC UNIT (ALU))

- یکی از بخش های مهم هر CPU واحد ALU است.
- این بخش محاسبات و عملیات منطقی را انجام می دهد.
- بخش های دیگر کامپیوتر داده ها را تحویل این بخش می دهند و نتیجه را دریافت می کنند. (واحد ALU مداری است که عملیات ریاضی را انجام می دهد.)
- در حالت پایه تنها با مقدارهای صحیح (integer) کار می کند.
- بعضی از ALU ها قابلیت کار با اعداد حقیقی (floating point) را نیز دارند.



نمایش اعداد صحیح (بدون علامت)

```
00000000 =   0
00000001 =   1
00101001 =  41
10000000 = 128
11111111 = 255
```

- در کامپیوتر همه اعداد در مبنای ۲ (باینری) ذخیره می شوند.
- در مبنای دو دو رقم بیشتر نداریم: 0 و 1
- مثال هایی از تبدیل اعداد مبنای ۱۰ به ۲:
- دقت کنید که این اعداد این مثال بدون علامت هستند: یعنی درباره مثبت و منفی بودن عدد هنوز صحبت نکرده ایم.
- هر مکان اعداد در مبنای ۲ دارای ارزش 2^m است. m شماره مکان عدد از سمت راست است (اولین مکان صفر است.).
- در اعداد بی علامت:
  - جمع دو عدد n بیتی نتیجه n بیتی به اضافه یک بیت carry خواهد داشت.
  - برای افزایش تعداد بیتهای عدد کافیست تا از سمت چپ به آن صفر اضافه کنیم.

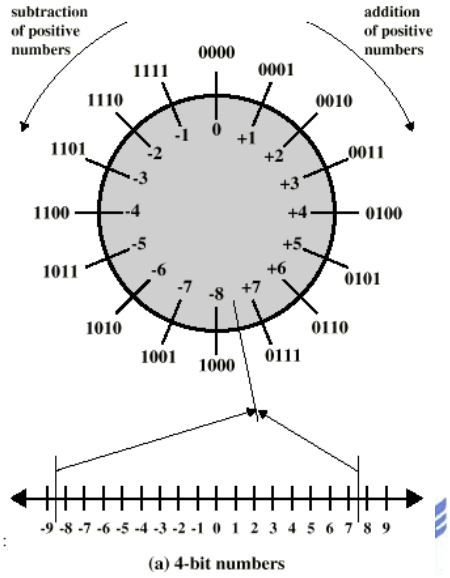
برای نمایش اعداد منفی در کامپیوتر از سیستمی با نام مکمل ۲ استفاده می شود. (به این روش نمایش می گویند: اعداد علامت دار.)

- برای منفی کردن هر عدد را مکمل ۲ کنید.
  - مکمل ۲ کردن: تمام ارقام را برعکس کنید (0به 1 و 1 به0 تبدیل شوند)
  - سپس کل عدد را با 1 جمع کنید.
- در این روش محاسبات اعداد منفی و تفریق بسیار ساده می شود.
- نکته های مهم در نمایش اعداد علامت دار. بیت سمت چپ عدد نشان دهنده علامت عدد خواهد بود. (0 = مثبت و 1 = منفی)
  - جمع دو عدد علامت دار n بیتی، نتیجه ای n بیتی خواهد داشت. بیت carry در هر شرایط دور ریخته خواهد شد.
  - برای افزایش تعداد بیت هر عدد باید بیت سمت چپ را تکرار کنید. (افزودن 0 یا 1)
  - در جمع ممکن است سرریز اتفاق بیافتد. (اگر جمع دو عدد منفی مثبت شود یا جمع دو عدد مثبت منفی شود.)

مثال از نمایش اعداد علامت دار چهار بیتی

Decimal Representation	Twos Complement Representation
+8	—
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	—
−1	1111
−2	1110
−3	1101
−4	1100
−5	1011
−6	1010
−7	1001
−8	1000

- در این نوع نمایش یک حالت چرخشی وجود دارد یعنی اگر به 7+ یکی اضافه شود به 8- می رسیم. (سرریز اتفاق افتاده.)



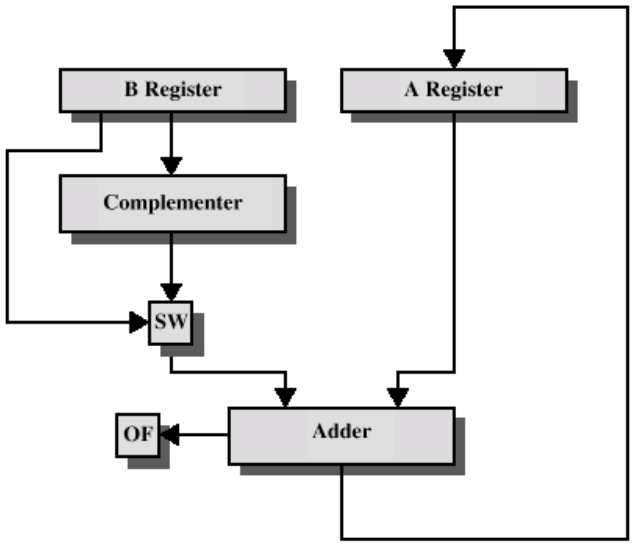
محدوده اعداد علامت دار و بی علامت
نحوه جمع و تفریق در سخت افزار

- محدوده اعداد قابل نمایش برای n بیت
  - اعداد بی علامت: 0 تا 2^n-1 (مثال: برای n=8 محدوده: 0 تا 255)
  - اعداد علامت دار: $-2^{(n-1)}$ تا $+2^{(n-1)}-1$ (برای n=8 محدوده: 128- تا 127+)
- اگر بخواهیم در سخت افزار جمع و تفریق را انجام دهیم چه کار باید انجام دهیم:
- جمع: بدون توجه به علامت جمع را به شکل عادی انجام می دهیم.
  - فقط باید مراقب سرریز باشیم. اگر سرریز اتفاق افتاد پاسخ اشتباه است و جمع باید با تعداد بیت بیشتری تکرار شود.
- تفریق: برای انجام a-b می نویسیم: a+(-b)
  - یعنی ابتدا از b مکمل دو می گیریم و جمع عادی انجام می دهیم.

- بنابرین برای این دو عمل تنها به جمع کننده و مکمل ۲ گیر نیاز داریم.

سخت افزار لازم برای جمع و تفریق

- مدار مقابل مقدار رجیستر A را با B جمع یا تفریق می کند.
- SW (که n مالتی پلکسر 2x1 است.) تعیین می کنند که جمع انجام شود یا تفریق.



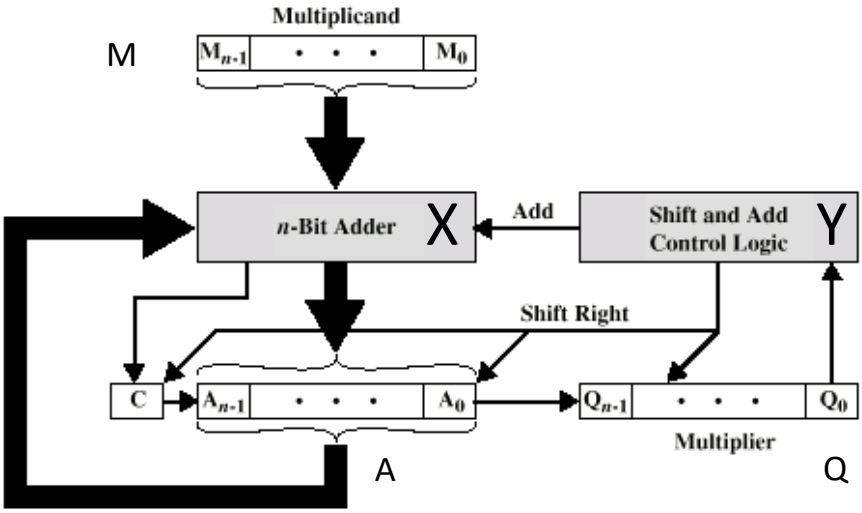
OF = overflow bit
SW = Switch (select addition or subtraction)

پیاده سازی سخت افزاری مدار ضرب کننده

```
    1011      Multiplicand (11)
  ×1101       Multiplier (13)
    1011   }
    0000   }  Partial products
   1011    }
  1011     }
10001111      Product (143)
```

- روش معمول ضرب دستی به کار گرفته می شود.
- فعلاً برای اعداد بی علامت صحبت می کنیم.
- دقت کنید که در ضرب مبنای دو هر سطر ضرب یا کلاً صفر است یا تکرار عدد ضرب شده اول است. (چون فقط اعداد صفر و یک داریم.)
- ضرب n بیت در n بیت نتیجه 2n بیتی خواهد داشت + یک بیت carry
- همین روند ضرب در مدار پیاده می شود.
- کافیست برمبنای بیت های عدد دوم مقدار عدد اول یا صفر را به ترتیب با هم جمع کنیم. البته جمع انجام شده باید هر بار با شیفت دادن (حرکت دادن) عدد دوم به چپ انجام شود.
- به این روش ضرب می گوییم شیفت و جمع.

مدار ضرب کننده بی علامت به روش شیفت و جمع



در این شکل سفید ها رجیستر هستند.
و خاکستری ها مدار منطقی هستند.
مدار X یک جمع کننده n بیتی است.
مدار X جمع را بین A و M انجام می دهد.
می توان به جای M صفر را انتخاب کرد.
مدار Y این انتخاب را انجام می دهد.
نتیجه جمع در A و C ذخیره می شود.
پس از هربار جمع کل Q و A و C یک بیت به سمت راست شیفت داده می شوند.

- ابتدا عدد اول و دوم طبق شکل بالا در دو رجیستر M و Q قرار می گیرد. رجیستر A صفر می شود.
- براساس بیت Q0 مدار Y تعیین می کند که ورودی مدار X صفر باشد یا مقدار رجیستر اول. نتیجه جمع در A و Cذخیره می شود.
- در پایان بیت های Q و A و C یک بیت به سمت راست شیفت داده می شوند.
- این کار n بار تکرار می شود تا همه بیت های Q به ترتیب وارد Y شوند.
- در پایان نتیجه ضرب در Q و A و C ذخیره خواهد شد. (و عدد Q اولیه از بین می رود.)

مثال از مراحل انجام شده ضرب دو عدد به روش جمع و شیفت
مقدارهای زیر محتوای رجیسترها را پس از انجام هر مرحله نشان می دهد.
در اینجا می خواستیم M و Q را در هم ضرب کنیم.
خط چین ها در شکل زیر مرز بین عدد Q و نتیجه را نشان می دهد.

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

ضرب اعداد علامت دار و تقسیم اعداد صحیح

- ضرب اعداد علامت دار به سادگی جمع تفریق اعداد علامت دار نیست.
- یعنی نمی توان دو عدد علامت دار را به روش عادی ضرب کرد و انتظار جواب درست داشت.
- مثال:

1001 (9) ×0011 (3) 00001001 1001 × 2^0 00010010 1001 × 2^1 00011011 (27)	1001 (−7) ×0011 (3) 11111001 (−7) × 2^0 = (−7) 11110010 (−7) × 2^1 = (−14) 11101011 (−21)

- راه حل

1. ابتدا اعداد منفی را مثبت کنیم (با مکمل ۲) در نهایت تاثیر علامت ها را روی نتیجه لحاظ کنیم.
2. الگوریتم Booth (برای مطالعه)

- تقسیم اعداد صحیح هم همانند روش دستی روی مدار پیاده سازی می شود. و برای اعداد علامت دار هم پیچیده تر می شود. (برای مطالعه)

نمایش اعداد اعشاری (حقیقی) در کامپیوتر

- اعداد اعشاری = اعداد صحیح + قسمت اعشار
- نمایش اعداد اعشاری در مبنای ۲ (برمبنای ارزش مکانی انجام می شود.)
- $1001.1010 = 2^3 + 2^0 + 2^{-1} + 2^{-3} = 9.625$
- برای تبدیل اعداد اعشاری مبنای ۱۰ به مبنای ۲ ضرب متوالی در ۲ انجام می دهیم و هر بار مقدار صحیح بدست آمده از نتیجه را حذف می کنیم.
- نحوه نگه داری در کامپیوتر
  - ممیز ثابت: تعداد مشخصی بیت به قبل و تعداد مشخصی بیت به بعد از ممیز اختصاص دهیم. که این کار محدوده اعداد کوچکی را ایجاد می کند. این روش استفاده نمی شود.
  - ممیز شناور: اسلاید بعد

روش نگهداری ممیز شناور (FLOATING POINT) اعداد طبیعی

- ممیز شناور: ابتدا عدد را به فرم علمی در مبنای ۲ در میاوریم و پایه و نمای این عدد را ذخیره می کنیم. این روش غالب نگهداری اعداد اعشاری در کامپیوتر است.
- $1001.1010 = 1.001101 \times 2^3$
- برای این عدد 001101 را به عنوان پایه و 3 را به عنوان نما ذخیره می کنیم.
- دقت کنید که 1 سمت چپ ممیز ذخیره نمی شود چون همیشه مقدار اول عدد یک است.
- و البته واضح است که 3 به صورت باینری (11) ذخیره خواهد شد
- در شکل استاندارد ۲۳ بیت برای پایه و ۸ بیت برای نما در نظر گرفته شده.
- یک بیت هم علامت را نشان می دهد. $+/- .significand \times 2^{exponent}$

(a) Format

مثال از نگهداری چند عدد طبیعی به روش ممیز شناور

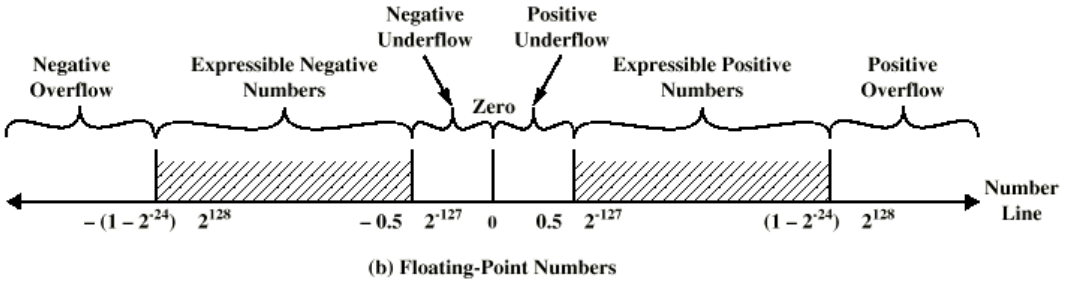
- نما به روش خاصی ذخیره می شود. نما ابتدا با ۱۲۷ جمع می شود و ذخیره می شود.
- مثال: اگر نما ۳ باشد، ابتدا با ۱۲۷ جمع می شود و ۱۳۰ ذخیره می شود.
- در مثال زیر ۲۰ با ۱۲۷ جمع شده و عدد ۱۴۷(10010011) ذخیره شده است.
- همچنین ۲۰- با ۱۲۷ ذخیره شده و ۱۰۷ (01101011) ذخیره شده است.
- این روش ذخیره سازی محاسبات را ساده می کند.
- به این ترتیب حداقل نما ۱۲۷- و حداکثر آن ۱۲۸ می شود.

(a) Format

$1.1010001 \times 2^{10100} = 0\ 10010011\ 10100010000000000000000 = 1.638125 \times 2^{20}$
$-1.1010001 \times 2^{10100} = 1\ 10010011\ 10100010000000000000000 = -1.638125 \times 2^{20}$
$1.1010001 \times 2^{-10100} = 0\ 01101011\ 10100010000000000000000 = 1.638125 \times 2^{-20}$
$-1.1010001 \times 2^{-10100} = 1\ 01101011\ 10100010000000000000000 = -1.638125 \times 2^{-20}$

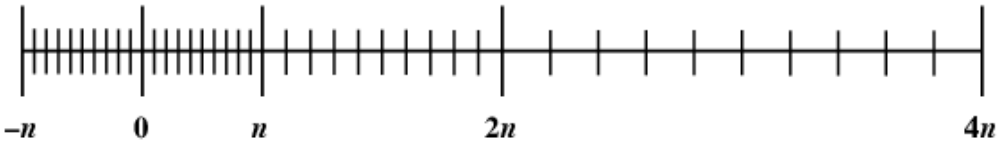
محدوده اعداد با روش نمایش ممیز شناور

- نسبت بین بزرگترین و کوچکترین عدد = $2^{256} \approx 1.5 \times 10^{77}$
- دقت عدد نمایش داده شده = $2^{-23} \approx 1.2 \times 10^{-7}$
- یعنی اعداد در مبنای ده با حدود ۶ تا ۷ رقم بعد از اعشار نگه داری می شوند.
- در شکل زیر محدوده اعداد قابل نمایش با خط چین نشان داده شده است.



(b) Floating-Point Numbers

- فاصله بین اعداد نمایش داده شده در اعداد کوچک کمتر می شود:

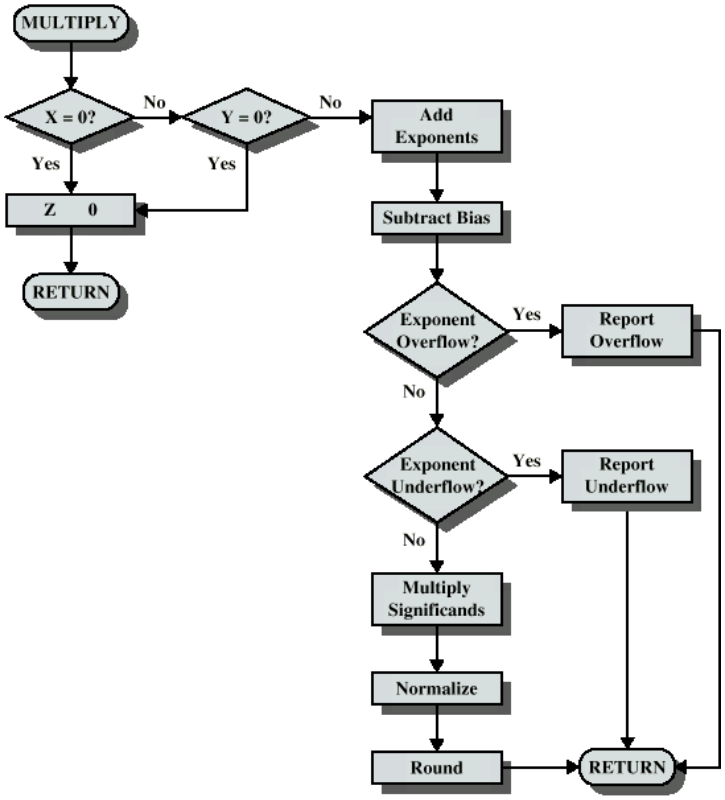


عملیات ریاضی روی اعداد ممیز شناور

- در اعداد ممیز شناور ضرب و تقسیم بسیار راحتتر از جمع و تفریق انجام می شود.
- مراحل انجام ضرب و تقسیم (روند ضرب و تقسیم مشابه است و با یک مدار انجام می شود.)

1. صفر بودن اعداد بررسی می شود. (در ضرب نتیجه صفر می شود. در تقسیم نتیجه یا صفر می شود و یا خطا گزارش می شود.)
2. نماها با هم جمع می شوند (ضرب) یا از هم تفریق می شوند (تقسیم).
3. پایه ها ضرب یا تقسیم می شوند. (با اعمال علامت ها)
4. نتیجه باید نرمال سازی شود. (یعنی باید پایه ها عددی بین یک و دو شود.)
   - در حین نرمال سازی مقدار لازم به نما اضافه می شود یا کم می شود.
5. رند کردن: تعداد رقم های پایه باید کم شود. (ضرب دو n بیت دو n بیت می شود ولی می خواهیم نتیجه همان n بیت بماند پس n بیت کم ارزش تر دور ریخته می شود.)
6. تمام محاسبات باید در 2n بیت انجام شود و روی نتیجه نهایی رند کردن انجام شود.

روند انجام ضرب دو عدد ممیز شناور در سخت افزار



- در ضرب اعداد ممیز شناور ممکن است overflow یا underflow رخ دهد.
- Overflow: ضرب دو عدد بیش از محدوده بالای اعداد شود.
- Underflow: ضرب دو عدد خیلی کوچک شود به طوری که از محدوده پایین خارج شود.
- در حالت اول خطا گزارش می شود.
- اکثراً در حالت دوم نتیجه صفر می شود.

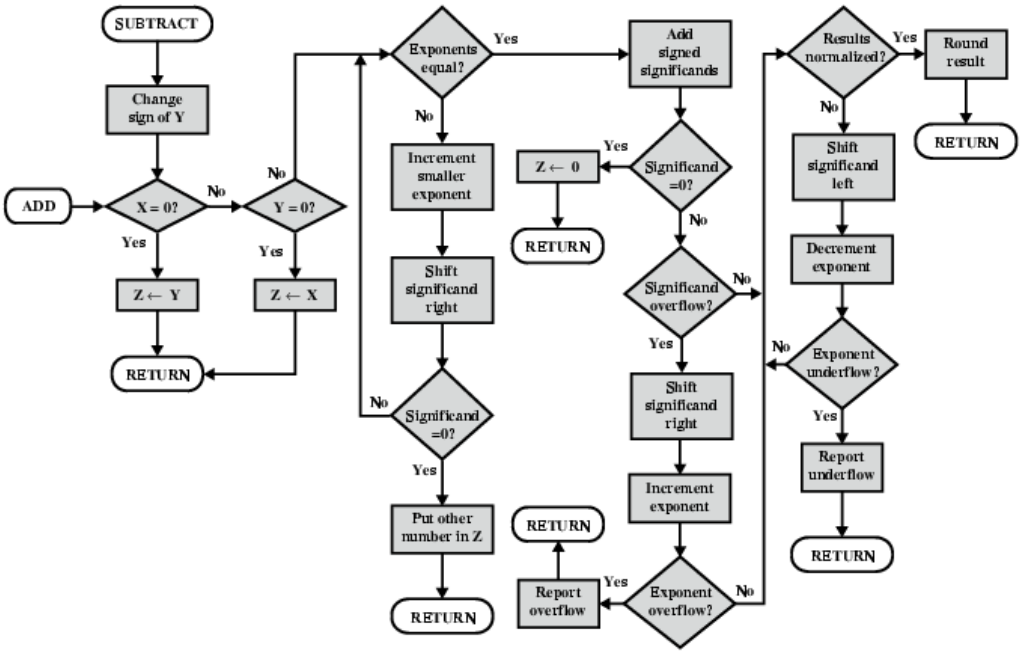
روند جمع و تفریق در اعداد ممیز شناور

- در اعداد ممیز شناور جمع و تفریق روندی مشکل تر دارد.
- ابتدا باید نمای دو عدد یکسان شود.
- این کار با شیفت دادن پایه عدد کوچکتر به راست انجام می شود.
- سپس پایه ها با هم جمع می شوند.
- در نهایت نرمال سازی نتیجه انجام می شود.
- نتیجه: همه اعمال ممیز شناور با همان مدارهای جمع و ضرب ساخته می شود.
- مثالی از چهار عمل اصلی در مبنای ۱۰: (در کامپیوتر همه عملیات در مبنای دو انجام می شود.)

$$X = 0.3 \times 10^2 = 30$$
$$Y = 0.2 \times 10^3 = 200$$

$$X + Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3 = 0.23 \times 10^3 = 230$$
$$X - Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3 = (-0.17) \times 10^3 = -170$$
$$X \times Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^5 = 6000$$
$$X \div Y = (0.3 \div 0.2) \times 10^{2-3} = 1.5 \times 10^{-1} = 0.15$$

روند انجام جمع و تفریق دو عدد ممیز شناور در سخت افزار (برای مطالعه)



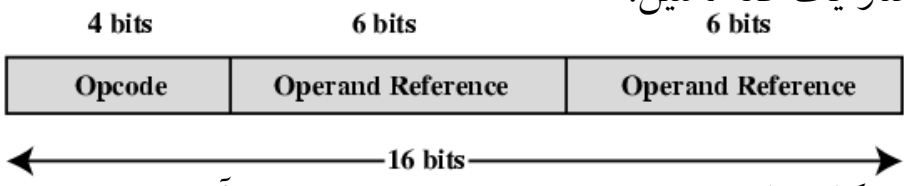
ساختار دستورها: خصوصیات و کارکردها

- در اینجا می خواهیم مجموعه دستورهای کامپیوترها را معرفی کنیم.
- مجموعه دستورها: همه دستورهایی که برای CPU تعریف شده است.
- دستور: (یادآوری) یک کد (عدد) باینری (معروف به کد ماشین) است که از حافظه برداشته می شود (fetch) و به رجیستر IR انتقال داده می شود. بر مبنای این مقدار تصمیم گیری (decode) می شود که CPU در این سیکل اجرای دستور باید چه کاری انجام دهد (execute).
- برنامه مجموعه ای از این کدهای ماشین است. هر کد ماشین یکتا است.
- در زبان اسمبلی برای هر کدماشین یک نماد در نظر گرفته می شود. (برای راحت تر نوشته شدن و مفهوم بودن برای کاربر انسانی) مثال: ADD، SUB، LOAD
- در این بخش می خواهیم ببینیم که این دستور (کد ماشین) چه ساختاری دارد.
- چهار نوع دستور داریم: پردازش داده، ذخیره و بازیابی داده از حافظه، ذخیره و بازیابی داده از دستگاه های I/O و کنترل جریان برنامه.

دستور (کد ماشین) دارای چند بخش اصلی است.
بخش اول اجباری است و بقیه بخش ها بستگی به نوع دستور دارند.

1. کد عملیات (Operation Code) معروف به OP code
   - نشان می دهد که این دستور چه کاری باید انجام دهد.
2. ارجاع به operand (عملگر) منبع: دستور باید روی این مکان اجرا شود.
3. ارجاع به operand (عملگر) مقصد: نتیجه باید در این مکان ذخیره شود.
4. ارجاع به دستور بعدی.
   - زمانی که این اجرای دستور تمام شد، دستور بعدی از این مکان برداشته می شود.

- مثالی از ساختار یک کد ماشین:



- دقت کنید که مکان ها همیشه به صورت صریح در دستور آورده نمی شود.
- مکان: جایی درون حافظه اصلی، رجیسترها، یا آدرس دستگاه های I/O.

برای دستورها چند ارجاع (قسمتهای بعد از OP CODE) بگذاریم؟

- دستورهای با سه ارجاع: دو مبدا، یک مقصد.
  - مثال: a = b + c;
  - ممکن است ارجاع چهارمی هم باشد برای دستور بعدی (معمولاً به کار نمی رود.)
  - در این روش از سه مکان مختلف استفاده شده یعنی محاسبه سه آدرس مختلف.
  - دستور سه ارجاع معمول نیست. چون طول دستور زیاد می شود. و زمان اجرا زیاد می شود.
- دستورهای با دو ارجاع: دو مبدا، یک مقصد. (مقصد برابر با یکی از مبداها است.)
  - مثال: a = a + b;
  - طول دستور کاهش پیدا می کند. (تعداد محاسبه آدرس و زمان اجرا کمتر می شود.)
  - برنامه نویسی سخت تر می شود. (دقت کنید دیگر به فرم سه ارجاع دستوری نداریم.)
  - معمول ترین روش دستورهای کامپیوترها همین است.

برای دستورها چند ارجاع (قسمتهای بعد از OP CODE) بگذاریم؟ (ادامه)

- دستورهای با یک ارجاع: یک مبدا، یک مقصد ثابت (معمولا رجیستری به نام AC)
  - مثال: AC = AC + b
  - دقت کنید در این روش همه عملیات روی یک رجیستر ثابت انجام می شود.
  - بنابرین در کدماشین کافیست تنها آدرس b آورده شود.
  - دستورها کوچکتر و زمان اجرای آنها سریعتر می شود.
  - کامپیوترهای اولیه دستورهایی به این شکل داشتند.
  - برنامه نویسی بسیار سخت تر می شود.
- دستورهای بدون ارجاع:
  - این دستورها با دسترسی به stack کار می کنند.
  - باید پیش از اجرای دستورها مقدارها داخل فضایی به نام stack قرار گیرد.
  - این روش معمول نیست.

مثال: مقایسه برنامه سه کامپیوتر با تعداد ارجاع های متفاوت.

$$Y = \frac{A - B}{C + (D \times E)}$$

- می خواهیم عبارت مقابل را محاسبه کنیم.
- در سه کامپیوتر مختلف برنامه نوشته شده است:
- هر چه تعدا ارجاع ها کمتر شود:
  - برنامه طولانی تری نیاز خواهد بود.

Instruction		Comment
SUB	Y, A, B	Y ← A − B
MPY	T, D, E	T ← D × E
ADD	T, T, C	T ← T + C
DIV	Y, Y, T	Y ← Y ÷ T

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	Y ← A
SUB	Y, B	Y ← Y − B
MOVE	T, D	T ← D
MPY	T, E	T ← T × E
ADD	T, C	T ← T + C
DIV	Y, T	Y ← Y ÷ T

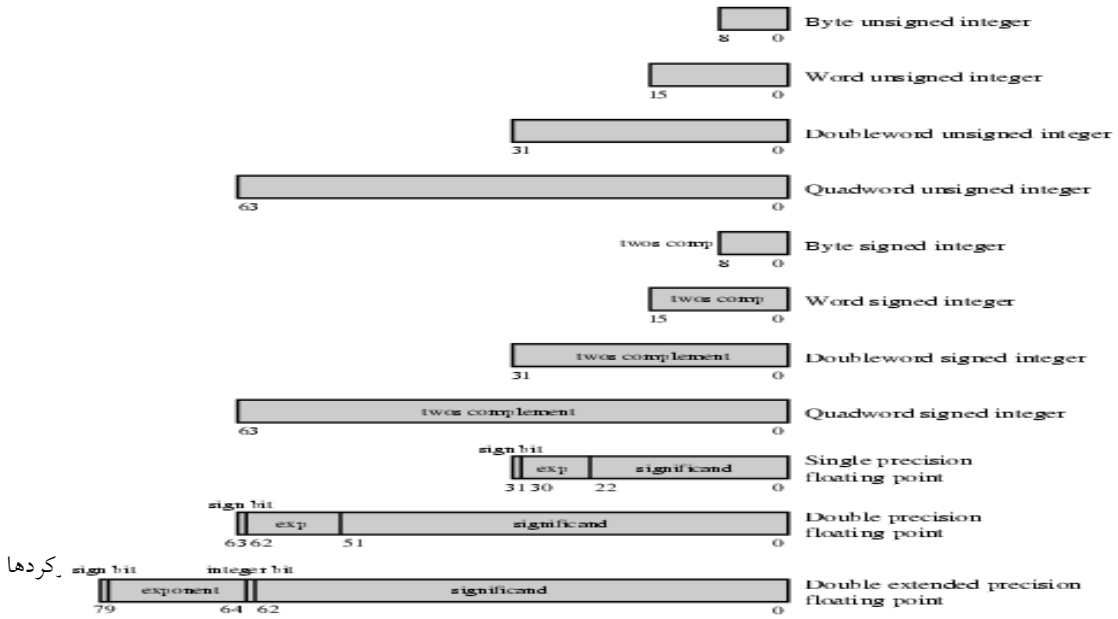
(b) Two-address instructions

Instruction		Comment
LOAD	D	AC ← D
MPY	E	AC ← AC × E
ADD	C	AC ← AC + C
STOR	Y	Y ← AC
LOAD	A	AC ← A
SUB	B	AC ← AC − B
DIV	Y	AC ← AC ÷ Y
STOR	Y	Y ← AC

(c) One-address instructions

نحوه نگه داری داده ها در کامپیوتر(انواع OPERAND ها)

- داده هایی که در کامپیوتر ذخیره می شود از چهار نوع است: آدرس، عدد (صحیح یا ممیز شناور)، کاراکترها، و داده های منطقی (بیت ها)
- دقت کنید که همه اینها به شکل باینری ذخیره می شود (صفر و یک). تفسیر ما به آنها معنا می بخشد.
- انواع داده های عددی در خانواده x86: (برای مطالعه)



کردها

دستورهای معمول در کامپیوترها

می توان دستورهای کامپیوترها را در چند گروه طبقه بندی کرد.

- Data Transfer (انتقال داده)
- Arithmetic (ریاضی)
- Logical (منطقی)
- Conversion (تبدیل)
- I/O (ورودی/ خروجی)
- System Control (کنترل سیستم)
- Transfer of Control (انتقال کنترل)
- در ادامه مثال هایی از هر گروه را مطرح می کنیم.