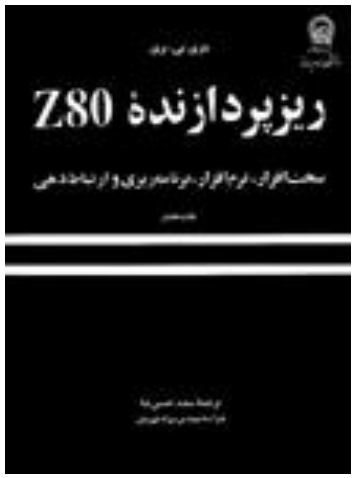
ریزپردازنده

MICROPROCESSOR

Books

- ***The Z80 Microprocessor , Hardware , Software programming & interfacing***
- ***Author:*** *Burry B. Brey*
- ***Translator:*** *Hossein Nia*
- **Publisher:** Astane Ghodse Razavi(Beh Nashr



Books

Microcompiuter and Microprocessor : the 8080, 8085, Z-80 Programi interfacing and trubleshooting

Publisher: Nass •
Pub.Date: 1381
Edition Turn: 3
ISBN: 964-6264-43-4-3
Pages: 719
Author: John E . Uffenbeck
Translator: Mahmmod Dayani •

Evolution of Computers

- First generation (1939-1954) - vacuum tube
- Second generation (1954-1959) - transistor
- Third generation (1959-1971) - IC
- Fourth generation (1971-present) - microprocessor

Integration levels

* **SSI** (small scale integration)
» Introduced in late 1960s
» 1-10 gates

* **MSI** (medium scale integration)
» Introduced in late 1960s
» 10-100 gates

* **LSI** (large scale integration)
» Introduced in early 1970s
» 100-10,000 gates

* **VLSI** (very large scale integration)
» Introduced in late 1970s
» More than 10,000 gates

مقدمه

- با پیدایش و پیشرفت مدارهای مجتمع کم کم این گرایش بوجود آمد که کامپیوترهای *کوچک و ارزان* ساخته شود.

- پیشرفت تکنولوژی ساخت IC ها و پیدایش تکنولوژی MOS در دهه ۶۰ امکان ساخت مدارهای مجتمع با مقیاس بزرگ (LSI) را در اوایل دهه ۷۰ بوجود آورد.

- استفاده از تکنولوژی LSI باعث میشد که بتوان یک پردازنده (processor) را بطور کامل به صورت یک تراشه ساخت .

- پردازنده بخش مرکزی یک کامپیوتر است که با گرفتن دستورات مختلف میتواند عملیات جمع و تفریق و مقایسه و یا عملیات منطقی را انجام دهد.

مقدمه ادامه:

- پردازنده هایی که بصورت تک تراشه ساخته میشوند بدلیل اندازه کوچکی که دارند ریزپردازنده (microprocessor) نامیده میشوند.

- اولین ریزپردازنده ایی که به بازار آمد تراشه ۴۰۰۴ شرکت اینتل بود که بعنوان یک تراشه قابل برنامه ریزی بروی ماشین های حساب طراحی گردید.

- این تراشه بدلیل قابلیت برنامه ریزی که داشت توانست به سرعت در بخش های مختلف صنعتی بعنوان بخش اصلی بعضی از سیستمهای کنترلی مورد استفاده قرارگیرد.

- استفاده از یک ریزپردازنده باعث میشود که بتوان از یک تراشه تنها با تغییر برنامه آن در بخشهای مختلف استفاده نمود.

- ریزپردازنده جای خود را در صنعت (مانند کنترل ماشین های صنعتی) و در مصارف خانگی ، علمی و آموزشی باز کرده است.

کاربرد ریزپردازنده ها:

۱- جایگزین نمودن سیستمهای دیجیتالی قدیمی بوسیله سیستم های مبتنی بر ریزپردازنده .

۲- ساختن کامپیوترهای بسیار کوچک (میکروکامپیوترها)

کاربرد ادامه:

- در طراحی یک سیستم دیجیتال، عمده ترین هدف به حداقل رساندن هزینه میباشد.
  این خود به حداقل رساندن تراشه هارا به دنبال خواهد داشت.

- برای به حداقل رساندن تعداد تراشه ها ، مناسبترین راه استفاده از تراشه های قابل برنامه ریزی میباشد.

- تراشه ها ی قابل برنامه ریزی :

۱- prom

۲- PLA

۳- microprocessors

uP به عنوان یک تراشه قابل برنامه ریزی :

- برنامه های مربوط به یک uP در یک یا چند تراشه حافظه که در کنار uP قرار دارند ذخیره میشود.

- دستورات هر برنامه متوالیا" توسط uP خوانده و به اجرا در میآید.

- با تغییر برنامه موجود در حافظه ، ازیک uP می توان در سیستم های مختلف استفاده نمود.
- استفاده از uP ها جهت ساختن سیستم های دیجیتالی ساده تر و سریعتر ازساخت سیستم های دیجیتال قدیمی است.

- زیرا مهارت های ساخت مدار های چاپی و لحیم کاری تراشه های مختلف بروی آنها جای خود را به وارد نمودن برنامه های نوشته شده به درون حافظه میدهد.

بزرگترین عیب سیتم های مبتنی بر ریزپردازنده سرعت کم آنها نسبت به سرعت سیستم های دیجیتال مشابه می باشد.

- تعداد تراشه های لازم برای ساخت یک ریز کامپیوتر :
  ۱- uP ۲- ROM ۳- RAM ۴- I/O

- استفاده از ریز کامپیوتر ها برای ساخت سیستم هایی که از حد مشخصی ساده تر هستند مقرون به صرفه نمی باشد.

- در این گونه موارد می توان از PROM و PLA و یا حتی از تراشه های MSI و SSI استفاده نمود.

اوایل دهه ۷۰ تکنولوژی LSI امکان ساخت ریزپردازنده ها را بوجود آورد.

- قراردادن واحد کنترل و واحد ریاضی و منطقی در یک تراشه باعث شکل گرفتن ریزپردازنده ها گردید.

- از نظر ساختمان داخلی ریزپردازنده ها را به سه دسته عمده تقسیم می کنند.

۱- ریزپردازنده های همه منظوره

General Purpose Microprocessors

۲- ریزکنترل کننده ها یا ریزکامپیوتر

Unit chip – Microcontrollers

۳- پردازنده های لایه ایی

Bit – Slice microprocessors

پارامترهای تعیین کننده قدرت یک ریز پردازنده:

۱- word size :

زمانی که طول ثباتها بزرگتر باشد، سرعت عملکرد پردازنده ها نیز بیشتر است.

مدیریت و کار با دستورالعمل های بزرگتر و اجزای اطلاعاتی بزرگتر ممکن می شود.

۲- تعداد خطوط گذرگاه داده :

از طریق گذرگاه داده است که داده ها می توانند از RAM به وسایل ورودی و خروجی و بلعکس انتقال داده شوند.

هرچه این مقدار بیشتر باشد سرعت کامپیوتر بیشتر است زیرا که هر بار اطلاعات بیشتری جهت پردازش بازیابی می شوند.

پارامترهای تعیین کننده قدرت یک ریز پردازنده: ادامه

۳ – تعداد خطوط گذرگاه آدرس :

بزرگ بودن گذرگاه آدرس بدین معنا است که مقدار حافظه قابل آدرس دهی پردازنده بیشتر است.

- قابلیت آدرس دهی حافظه های بزرگ دو مزیت مهم دارد.

۱- کاربرد برنامه های پیچیده و کارآمدتر آسان میشود.

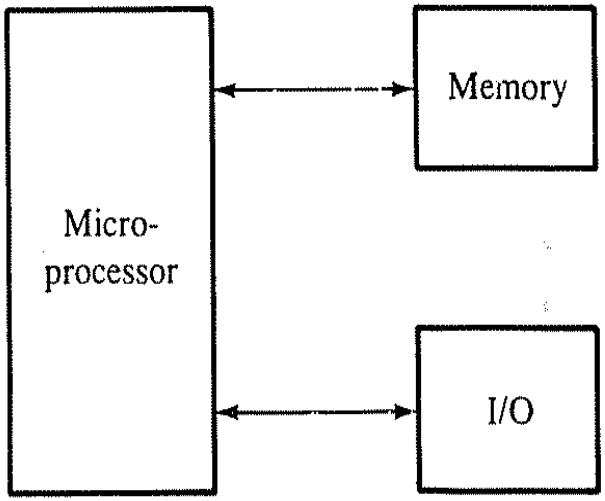
۲- Multitasking آسانتر میشود.

- **ریزپردازنده یک وسیله قابل برنامه ریزی منطقی چند منظوره بوده که دستورالعمل های باینری را از حافظه می خواند، داده های باینری را به عنوان ورودی دریافت و آنها را طبق دستورالعمل های موجود در حافظه پردازش و نتیجه را به صورت خروجی ارائه می نماید.**

یک ماشین قابل برنامه ریزی دارای سه قطعه اصلی می باشد:

ریزپردازنده، حافظه و ورودی/خروجی (I/O)

FIGURE 1.1
A Programmable Machine



- **وقتی یک سیستم مبتنی بر ریزپردزنده (سیتم فوق) طوری طراحی شود تا یک عمل خاصی را کنترل نماید به آن میکروکنترلر گویند و اگر برای پردازش داده و محاسبه بکار گرفته شود آن را میکروکامپیوتر می نامند.**

ارقام باینری:

- ریزپردازنده با اعداد باینری کار می کند که به آنها بیت گفته میشود. این ارقام بصورت ولتاژ الکتریکی مورد استفاده قرار می گیرند. صفر نشانگر یک ولتاژ است و یک نشانگر ولتاژ دیگری است. صفر به معنی ولتاژ پایین و یک به معنی ولتاژ بالا در نظر گرفته می شود.

- هر ریزپردازنده گروهی از بیت ها را تشخیص و مورد پردازش قرار می دهد که به آنها کلمه (Word) گفته می شود. ریزپردازنده ها بر اساس اندازه کلمه طبقه بندی می شوند.
- برای مثال ریزپردازنده ایی که با کلمه هشت بیتی کار کند ریزپردازنده ی ۸ بیتی نامیده می شود.

ریزپردازنده به عنوان یک وسیله قابل برنامه ریزی :

- ریزپردازنده یک وسیله برنامه پذیر است ، بدین معنی که می توان آنرا طوری هدایت نمود تا وظایف داده شده را در محدوده توانـایی اش انجـام دهـد. یعنـی ریزپردازنـده طوری طراحی گردیده است که می تواند دستورات باینری را درک و اجرا نماید.

حافظه

- حافظه مانند یک صفحه یا صفحه ها یی از یـک دفترچـه یاداشـت بـوده کـه دارای تعدادی خطوط ثابت برای نوشتن اعداد باینری بر روی هر خط آن می باشد. اگر چـه این صفحات از مواد نیمه هادی (semiconductors)ساخته شده اند. معمـولا هـر خط مانند یک ثبات هشت بیتی می باشد که می تواند ۸ بیت را در خود جای دهـد. و تعدادی از این ثبات ها که پشت سرهم قرار می گیرند حافظه را بوجود میآورند.

- کاربر دستورات و داده های لازم را از طریق وسـایل ورودی در حافظـه قـرار داده و از ریزپردازنده می خواهد که با اجرای آنها نتیجه را پیدا کند. نتیجـه معمـولا از طریـق وسیله خروجی به نمایش یا در حافظه ذخیره می گردد.

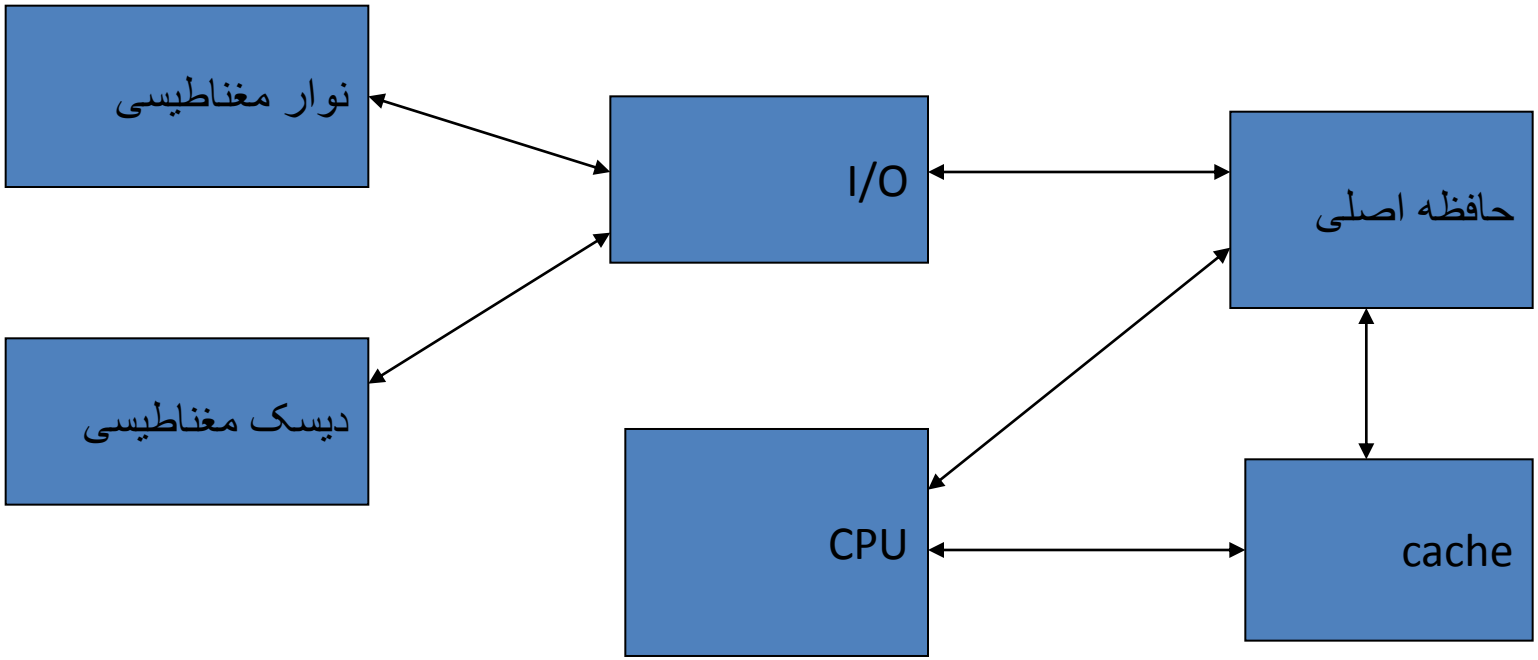
حافظه های سلسله مراتبی
memory hierarchy

- حافظه سلسله مراتبی شامل تمام وسایل ذخیره کننده اطلاعات سیستم کامپیوتری میباشد.

- از حافظه های کمکی با ظرفیت زیاد وسرعت کم تا حافظه اصلی نسبتا" سریع و بلاخره حافظه کوچکتر و بسیار سریع حافظه نهان (cache) میباشد.

- دلیل بکار بردن دو یا سه نوع حافظه با سرعت های مختلف یک مسله اقتصادی است.
- هرچه ظرفیت حافظه بیشتر باشد بهای واحد بیت آنها کمتر میشود و زمان دستیابی به اطلاعات افزایش می یابد.

حافظه های سلسله مراتبی



حافظه های سلسله مراتبی ادامه :

- هر چه سرعت حافظه بیشتر باشد، بهای واحد بیت آن نیز بالاتر است.

- حافظه نهان بخشی از برنامه و اطلاعات را ذخیره می کند که در جریان اجرای برنامه ، CPU بیشترین مراجعه به آنها را دارا است. در صورتی که حافظه های کمکی آن قسمت از برنامه را ذخیره می کنند، که فعلا مورد نیاز CPU نمیباشد.

- معمولا نسبت سرعت بین حافظه اصلی و حافظه نهان نسبت ۱ به ۷ است و زمان متوسط دسترسی به اطلاعات در حافظه های کمکی معمولا حدود ۱۰۰ برابر بیشتر از حافظه اصلی است.

- *هدف از حافظه های سلسله مراتبی این است که با حداقل هزینه حداکثر سرعت دستیابی به اطلاعات در یک سیستم کامپیوتری حاصل شود.*

حافظه memory

وظایف کلی حافظه عبارتند از :

۱- ذخیره نمودن برنامه.
۲- فراهم نمودن داده برای cpu در صورت درخواست.
۳- دریافت نتیجه از cpu جهت ذخیره نمودن.

حافظه ادامه:

- حافظه های یک میکرو کامپیوتر بر اساس کلمه (word) ساخته شده اند.
(1k کلمه یا 4k کلمه)

- برای دستیابی به محتوی یک حافظه به دو ثبات نیاز میباشد:

۱- MAR (memory address register) که دارای آدرس کلمه ایی است که نیاز به دست یافتن به آن میباشد.

۲- MDR (memory data register) که دارای داده ایی است که باید در داخل حافظه نوشته یا از داخل حافظه خوانده شود.

مثال : یک حافظه با ظرفیت 1k*8 دارای ۱۰۲۴ کلمه ۸ بتی میباشد.

حافظه ادامه:

بطور کلی حافظه به دو صورت عمل میکند:

۱- READ (خواندن) ۲- WRITE (نوشتن)

یک حافظه در حال خواندن است اگر اطلاعاتی در یک آدرس بخصوص در حافظه مورد نیاز باشد.

برای خواندن حافظه :

۱- آدرس محلی که باید داده از آن خوانده شود در MAR قرار میگیرد.

۲- فرمان Read توسط واحد کنترل صادر میشود.

۳- داده ایی که باید خوانده شود از حافظه به MDR منتقل میگردد که در این حالت داده از طریق گذرگاه داده در اختیار سیتم قرار میگیرد.

توجه: عمل خواندن باعث تخریب محتوی محل خوانده شده نمی شود.

حافظه ادامه:

برای نوشتن در حافظه مراحل زیر صورت میگیرد:

۱- آدرس محلی از حافظه که داده باید در آن نوشته شود در MAR قرار میگیرد

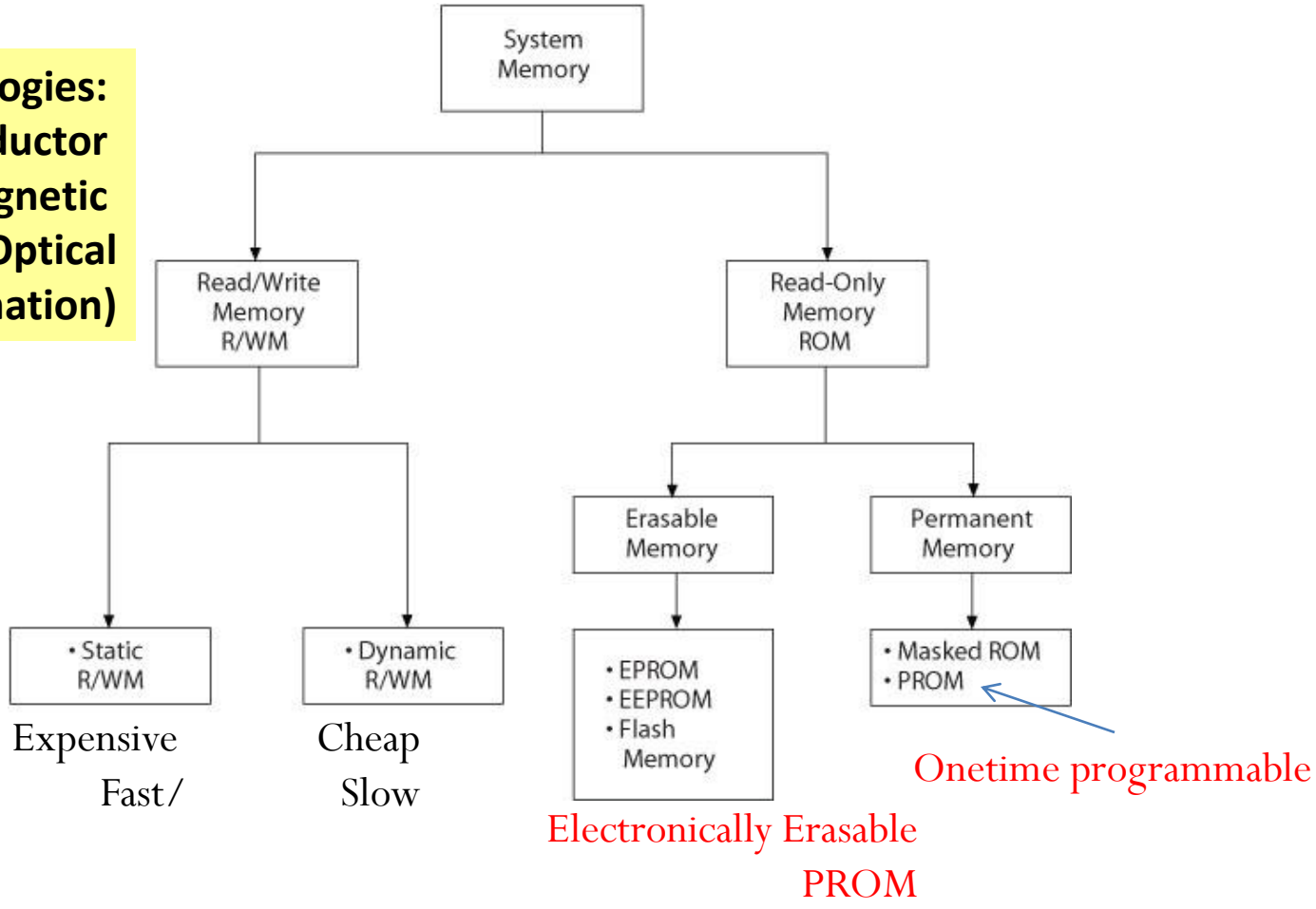
۲- داده ایی که باید نوشته شود در MDR قرار میگیرد.

۳- فرمان نوشتن از طریق خط R/W صادر میگردد و داده به محل آدرس شده منتقل میگردد.

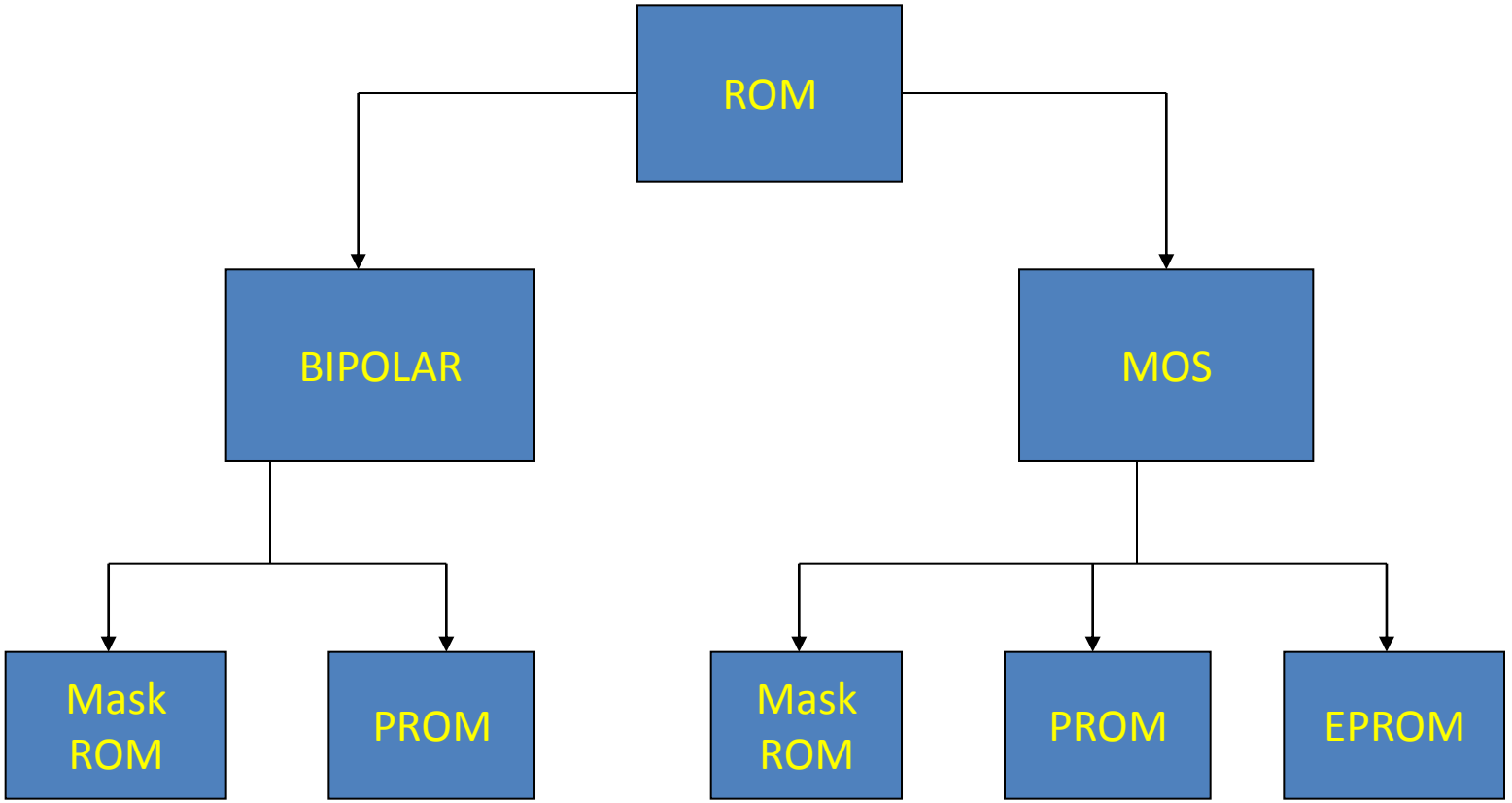
توجه: عمل نوشتن محتویات قبلی یک حافظه را از بین می برد.

Microprocessor-based Systems
Memory Classification

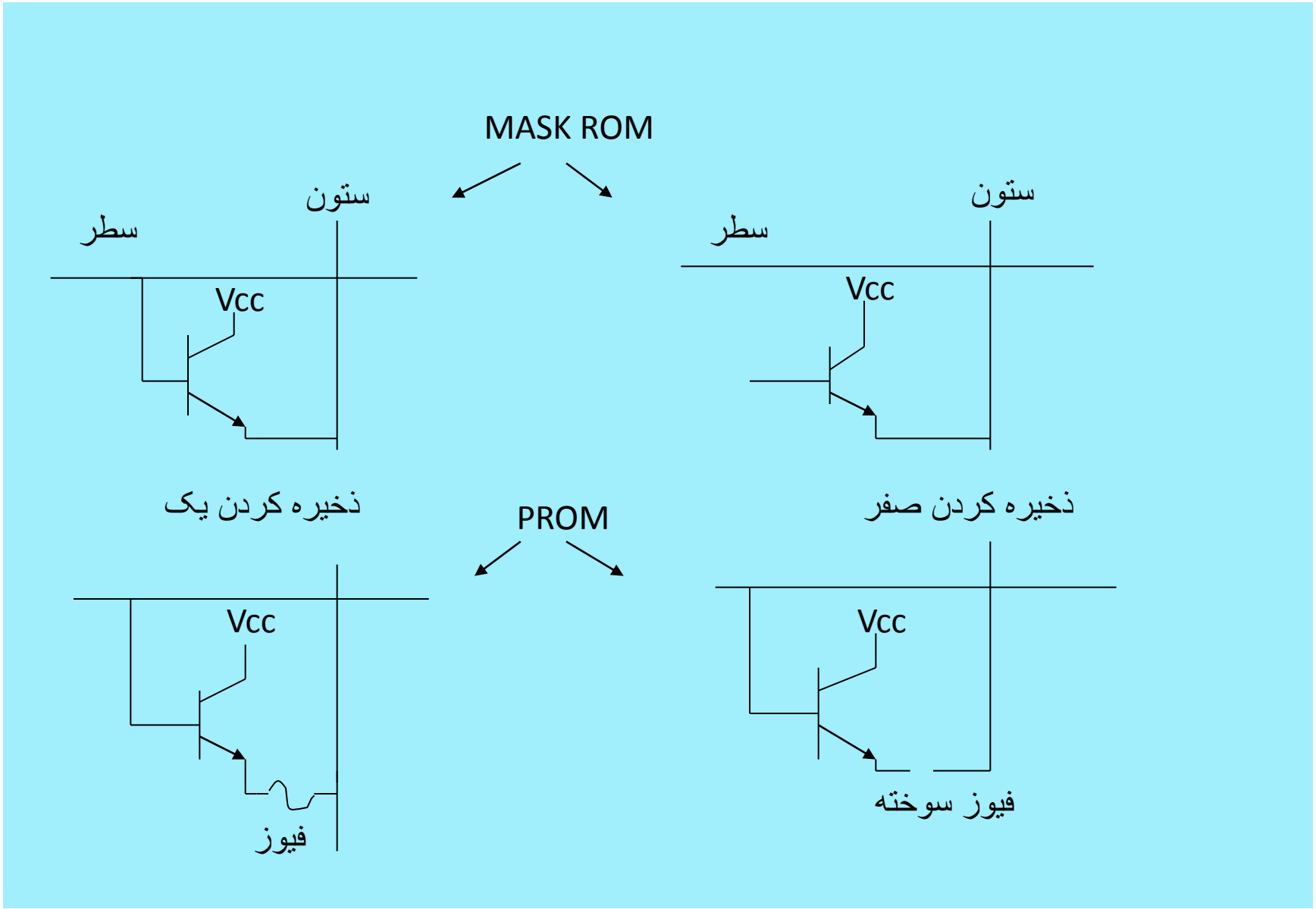
Basic Technologies:
Semiconductor
Magnetic
Optical
(or combination)



حافظه نیمه هادی ROM



حافظه ROM ادامه:



حافظه ROM ادامه :

یک ROM با ظرفیت ۱ کیلو بیت بصورت 256*4 را درنظر بگیرید.

در حالت معمولی ساختمان این حافظه به یک دیکودر 256*8 نیاز دارد و این حافظه از ۲۵۶ سطر (کلمه) و ۴ ستون (داده) تشکیل می گردد.

پیچیده گی (complexity) این حافظه از حاصل جمع خطوط خارج شده از دیکودر و تعداد ستونها (داده ها) بدست می اید. ۲۵۶ +۴ =۲۶۰

ROM عملی (ماتریسی)

- برای کاهش پیچیدگی و کاهش اندازه دیکودر به کار گرفته شده در یک حافظه ROM از طراحی ROM ماتریسی استفاده میشود.

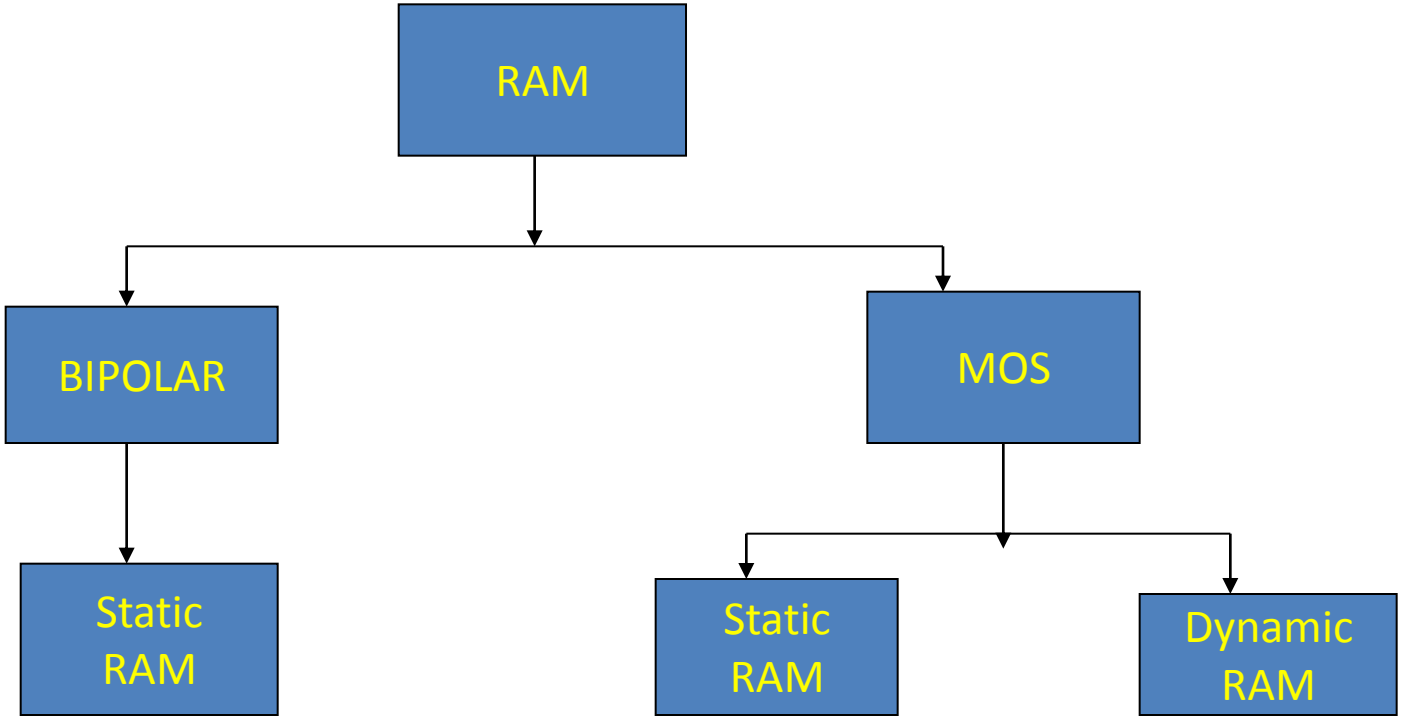
- یک ROM با ظرفیت ۱ کیلو بیت (256*4) را بصورت ماتریسی طراحی کنید.

در این ROM به وسایل زیر نیاز میباشد:

- یک مخزن 32*32
- یک دیکودر 5*32
- ۴ مالتی پلکسر 8*1

پیچیده گی این سیستم برابر است با ۶۴= ۳۲+۳۲

حافظه نیمه هادی RAM

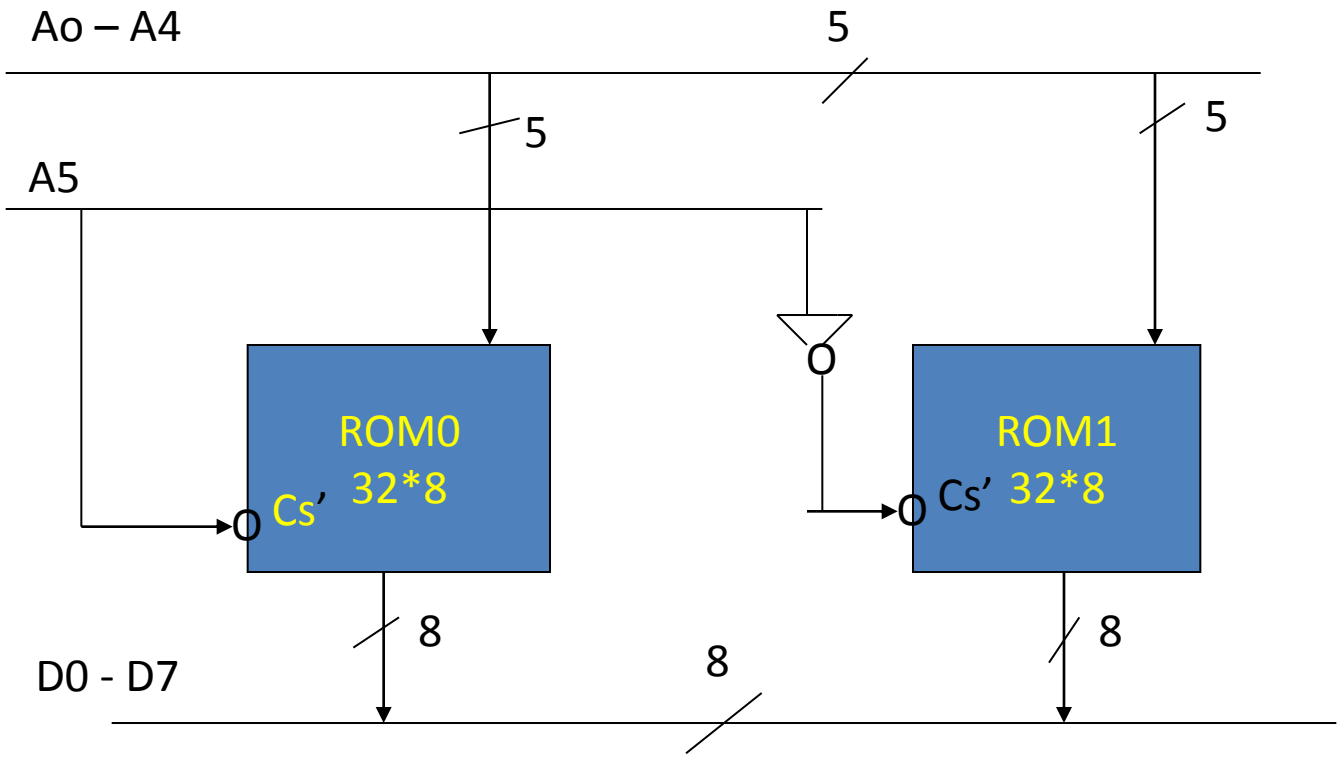


Memory expansion

- بسیار مطلوب می بود اگر تمام ظرفیت های مورد نیاز حافظه بصورت تراشه های مجزا موجود می بود.ولی چنین نیست:
- مثال:

اگر یک ROM 8*64 نیاز باشد، ولی فقط تراشه های ROM 8*32 موجود باشد با استفاده از دو تراشه 8*32 میتوان یک ROM 8*64 ساخت.

Memory expansion ادامه (افزایش تعداد کلمات حافظه)



Memory expansion ادامه

- اگر A5=0 باشد ROM0 انتخاب میشود .
- اگر A5=1 باشد ROM1 انتخاب میشود .

- محدوده آدرس ROM0 00h تا 1Fh (۳۲ بایت)
- محدوده آدرس ROM1 20h تا 3Fh (۳۲ بایت)
- محدوده کل حافظه ۶۴ بایتی 00h تا 3Fh میباشد.

- تمرین : یک ROM 8 * 128 را با استفاده از ROMهای 8*32 بسازید.

Memory expansion ادامه: (افزایش تعداد بیتهای هر محل حافظه)

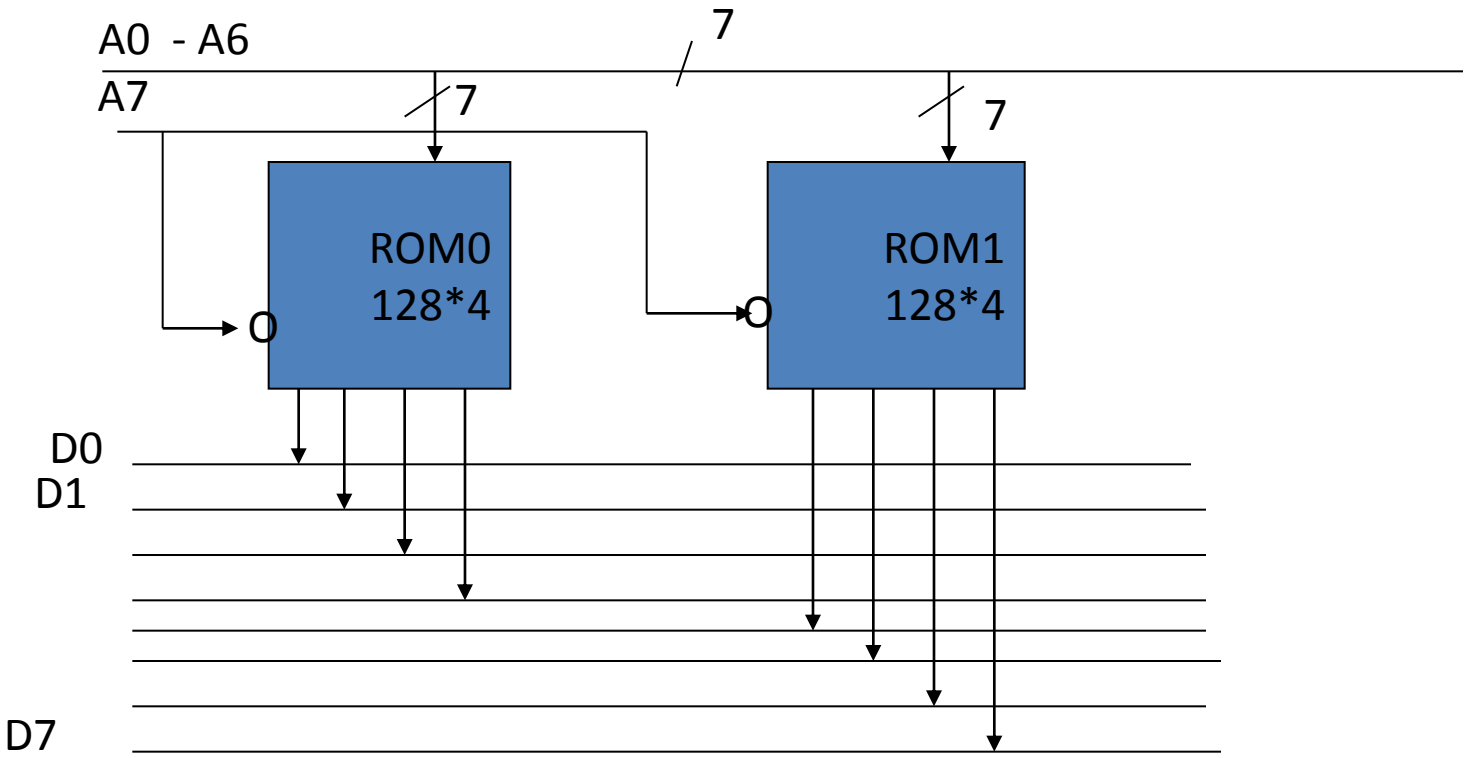
- گاهی اوقات حافظه هایی با طول کلمه ۸ بیتی ممکن است موجود نباشد.:

مثال:

یک ROM 128*8 نیاز بوده ولی فقط تراشه های 128*4 موجود می باشند.:

Memory expansion ادامه :

- اگر A7=0 باشد بطور همزمان محتوی آدرس شده هر یک از کلمات حافظه برروی گذرگاه داده قرار میگیرند.
- محدوده آدرس کل حافظه : 00h تا 7Fh



Memory address decoding

- یک ریزپردازنده ۸ بیتی معمولا دارای ۱۶ خط آدرس میباشد و در نتیجه می تواند به ۶۵۵۳۶ (۶۴ کیلو) محل از حافظه دسترسی داشته باشد.

- معمولا در یک عمل همه خطوط آدرس ممکن است لازم نباشد .

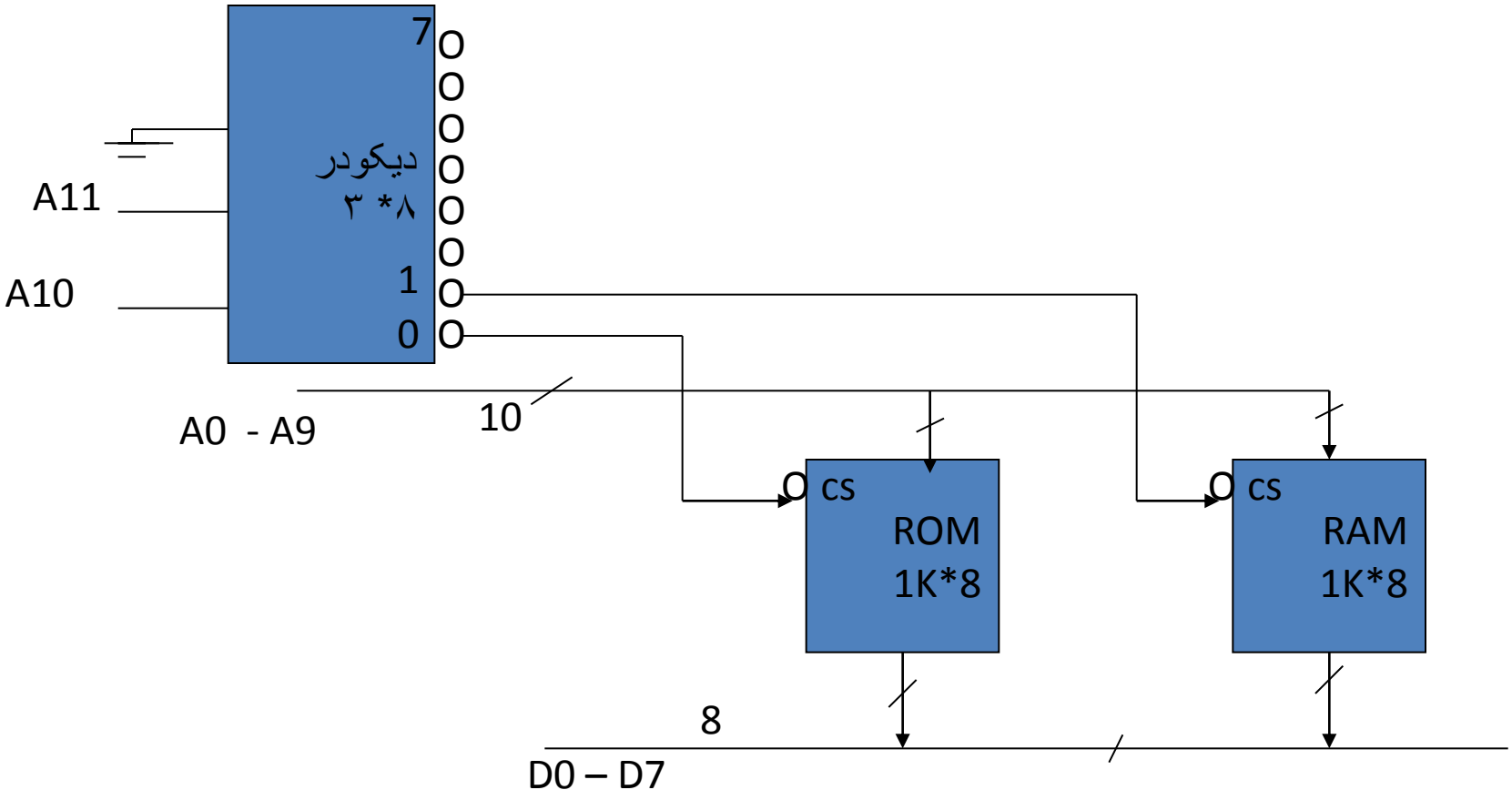
- حافظه مورد نیاز هم به صورت چندین تراشه مجزا بوده که باید از طریق CS هریک از تراشه ها به آنها دسترسی پیدا نمود.

- معمولا برای فعال نمودن csها از خطوط آدرس استفاده میشود.

- معمولا خطوط ارزش پایین گذرگاه آدرس را برای آدرس محل های حافظه و خطوط ارزش بالا گذرگاه آدرس را برای انتخاب تراشه ها استفاده میکنند.

Memory address decoding ادامه:

یک شیوه آدرس دهی نمودن یک حافظه ROM و RAM یک کیلو بایتی .



Memory address decoding ادامه:

- اگر A11=0 , A10=0 باشد حافظه ROM انتخاب میشود.
- اگر A11=0 , A10= 1 باشد حافظه RAM انتخاب میشود.

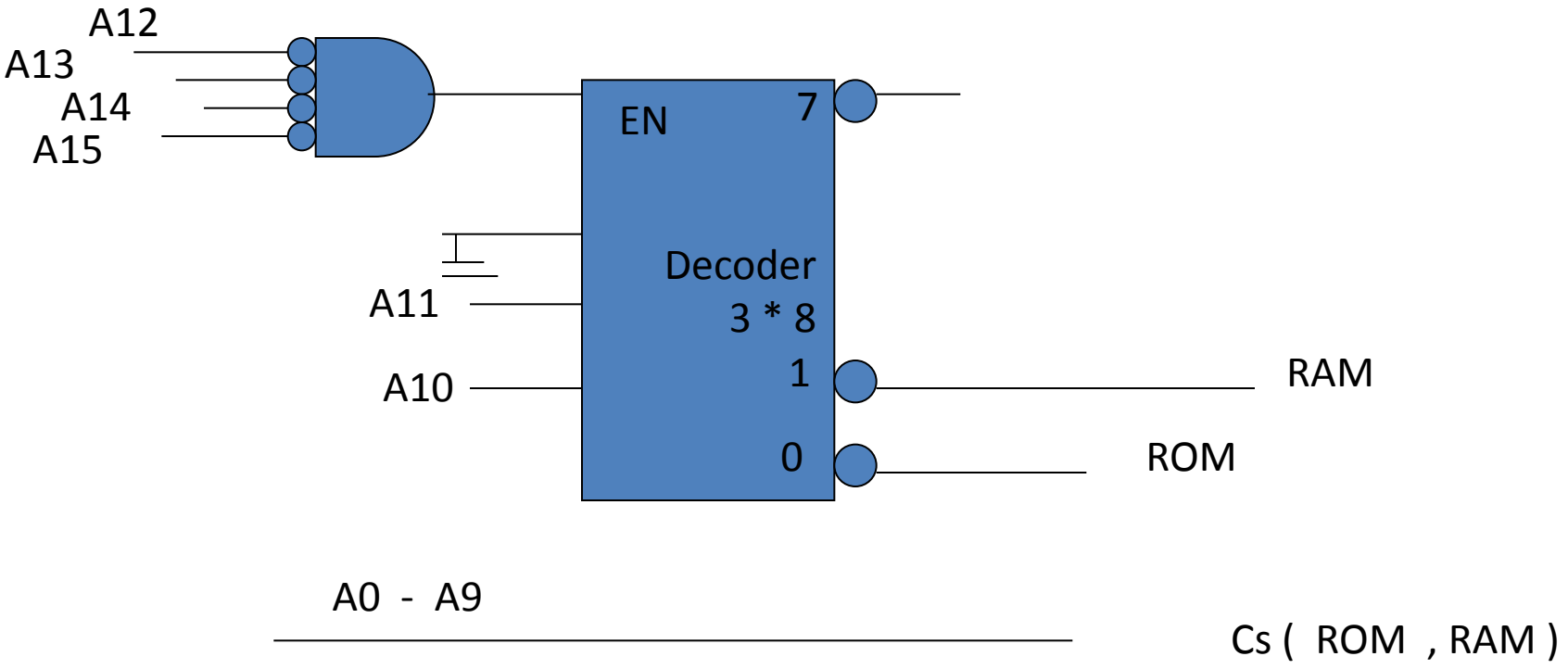
- محدوده آدرس ROM : 0000h تا 03FFh
- محدوده آدرس RAM : 0400h تا 07FFh

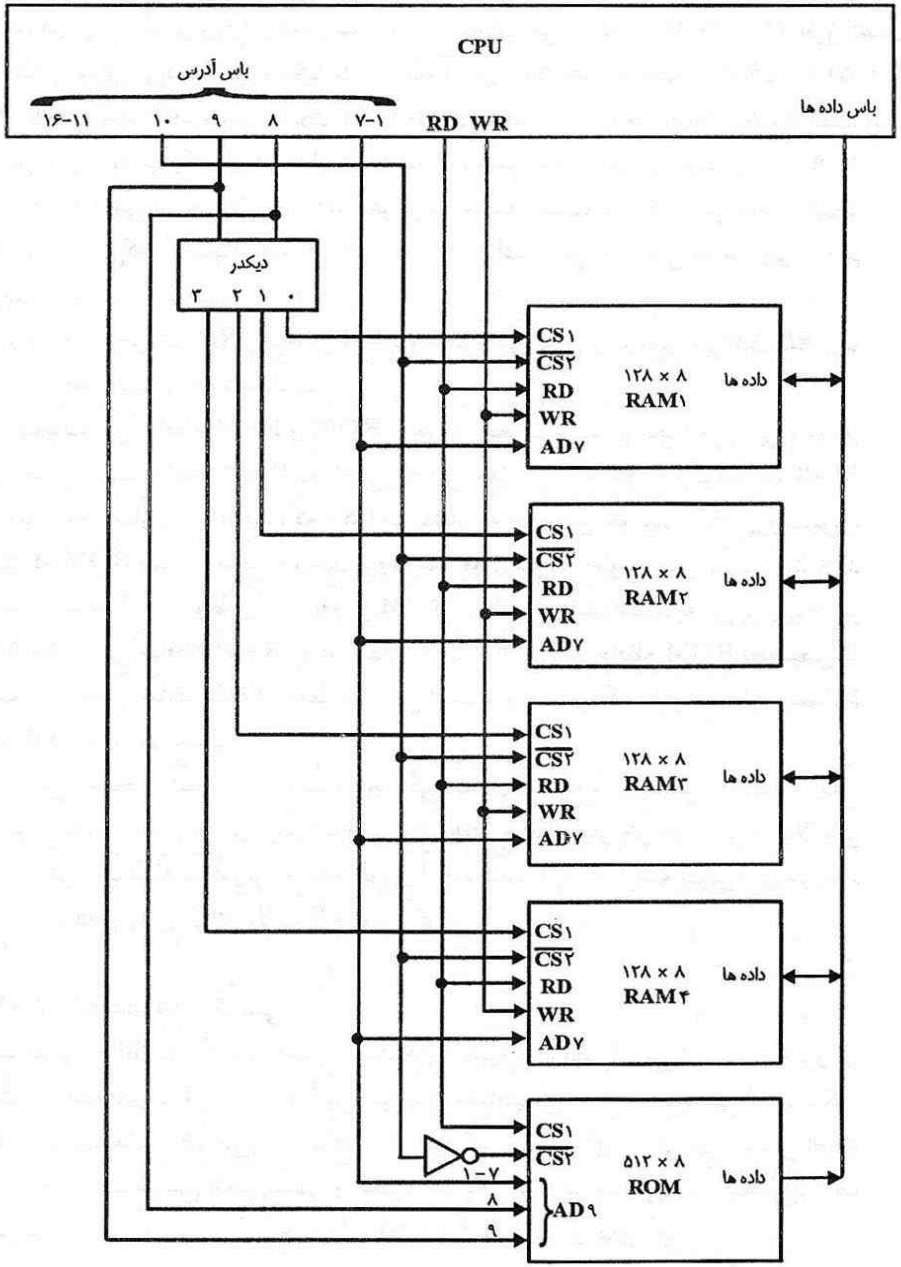
- از آنجاییکه چهار خط آدرس A12 تا A15 بجایی وصل نشده اند و میتوانند ۱۶ حالت مختلف بخود بگیرند، در نتیجه هر محل از حافظه ROM یا RAM می تواند توسط ۱۶ آدرس مختلف مورد دستیابی قرار گیرد.

  در این صورت برای هر محل حافظه یک آدرس ثابت وجود ندارد و به چنین روشی آدرس دهی non absolute addressing (آدرس دهی غیر مطلق)گفته میشود.

Memory address decoding ادامه:

آدرس دهی مطلق

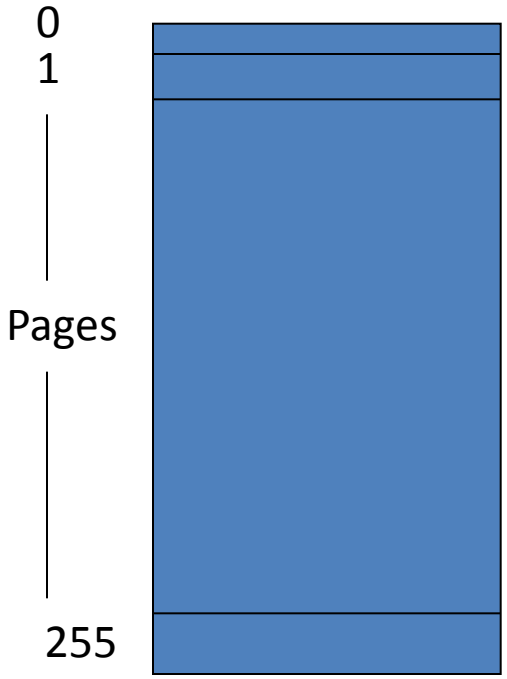




شکل (۱۲-۴) اتصال حافظه به CPU

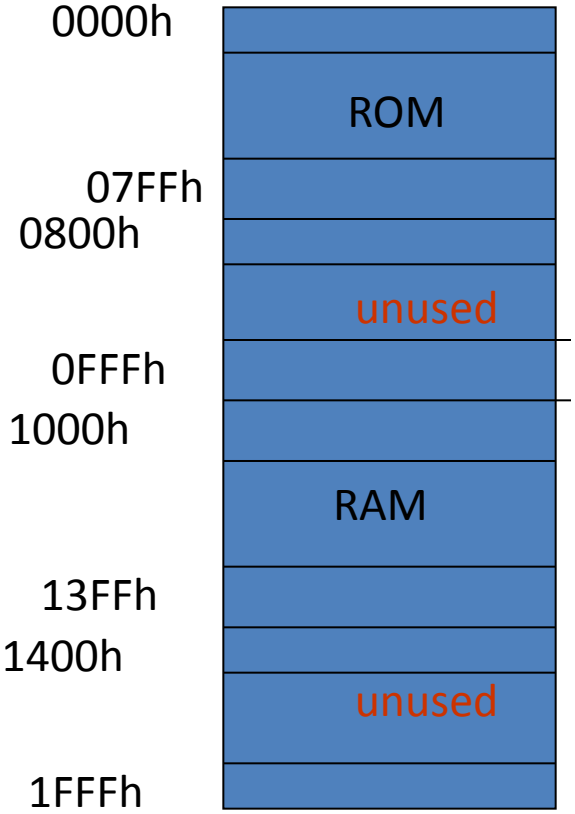
Memory pages
صفحات حافظه

- **در بعضی از ریزپردازنده ها که دارای ۱۶ خط آدرس میباشند، فضای آدرس دهی (۶۵۵۳۶محل) به ۲۵۶ بلوک (صفحه) که هر یک دارای ۲۵۶ محل (آدرس) می باشند تقسیم می شوند. (۶۵۵۳۶ = ۲۵۶ * ۲۵۶)**



دو رقم با ارزش بالا مشخص کننده صفحه می باشد.
دو رقم با ارزش پایین مشخص کننده محل حافظه می باشد.

Memory map
نقشه حافظه



- در یک سیستم مبتنی بر ریزپردازنده یک سازمان حافظه ۸ کیلو بایتی وجود دارد. در این سیستم یک حافظه ROM ۲ کیلو بایتی که از آدرس 0000h شروع میشود و یک حافظه RAM ۱کیلو بایتی که از آدرس 1000h شروع میشود وجود دارد. نقشه حافظه سیستم مورد نظر را ترسیم کنید.

Over flow

- در جمع دو عدد یک کلمه ایی ممکن است حاصل جمع قابل نمایش در یک کلمه نباشد . در این صوت ، حاصل را نمی توان در انباره که طول کلمه آن بیش از یک کلمه نیست نگهداری نمود که در این حالت می گوین سرریز (over flow) رخ داده است.

- بر حسب اینکه از چه روشی برای نمایش اعداد منفی استفاده میکنیم نتیجه را باید ترمیم کنیم.

- در ریزپردازنده ها پس از هر عمل جمع یا تفریق وقوع یا عدم وقوع سرریز در یک بیت ، که با حرف V نشان داده میشود ،ضبط می گردد.

قوانین حاکم بر جمع اعداد چند بیتی مکمل دو مورد بحث قرار میگیرند.
در جمع دو عدد هشت بیتی ۵ حالت متفاوت وجود داد.
۱- هر دو عدد مثبت و نتیجه حاصل قابل نمایش در یک کلمه میباشد.

۱۱۲ + ۷ = ۱۱۹ 01110000 +
00000111

01110111 C=0 و V=0

۲- هر دو عدد مثبت هستند ولی نتیجه حاصل قابل نمایش در انباره نیست.

۱۱۲ + ۴۸ = ۱۶۰ 01110000 +
00110000

10100000 , V = 1, C=0

نتیجه حاصل باید مثبت ۱۶۰ باشد در صورتی که ۹۶- است.
برای تصحیح نتیجه حاصل باید بیت نقلی را به عنوان بیت علامت در کنار انباره قرار داد تا نتیجه درست حاصل شود.

Over flow ادامه :

۳- هر دو عدد منفی ولی انباره سرریز نشده است:

-107 + -10 = -117

10010101 +
11110110

10001011 C=1 و V=0

۴- هر دو عدد منفی و نتیجه حاصل قابل نمایش در انباره نیست.

-111 + -39 = -150

10010001 +
11011001

01101010 C= 1 و V=1

Over flow ادامه :

۵- یک عدد مثبت ودیگری منفی است. در این حالت هرگز سرریز رخ نمی دهد.

A) عدد مثبت بزرگتر از قدر مطلق عدد منفی است.

-112 + 120 = 8 + 10010000

01111000

00001000 C= 1 و V= 0

B) عدد مثبت کوچکتر از قدر مطلق عدد منفی است.

-112 + 96 = -16 + 10010000

01100000

11110000 C= 0 و V= 0

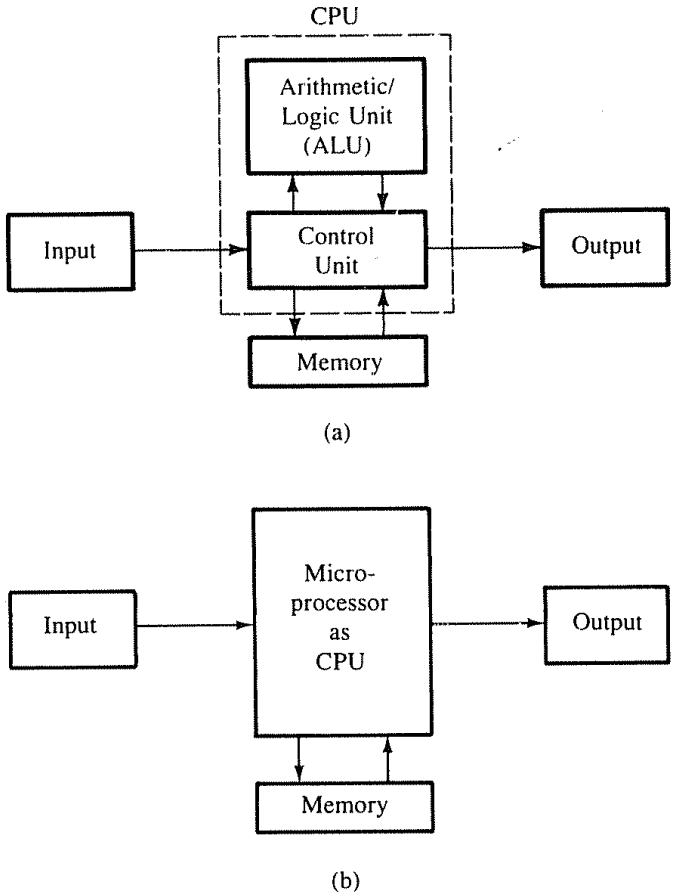
ورودی/خروجی

- کاربر می تواند دستورات و داده ها را از طریق وسیله ایی مانند یک صفحه کلید یا تعدادی کلید ساده وارد حافظه نماید. این وسایل، ورودی نامیده می شوند. ریزپردازنده دستورات را از حافظه خوانده و داده ها را طبق همان دستورات پردازش می نماید و نتیجه را توسط وسیله ای مانند یک نمایشگر هفت قسمنی و یا یک چاپگر به نمایش می گذارد. این وسایل را خروجی می نامند.

ریزپردازنده به عنوان یک واحد پردازش مرکزی(CPU):

بطور سنتی، کامپیوتر بصورت بلوک دیاگرام شکل 1.2(a) نمایش داده می شود.

FIGURE 1.2
(a) Traditional Block Diagram of a Computer (b) Block Diagram of a Computer with the Microprocessor as CPU



سازمان میکروکامپیوتر:

- شکل 1.3 ساختار ساده ولی رسمی یک ریزکامپیوتر(Microcomputer) را نشان میدهد.

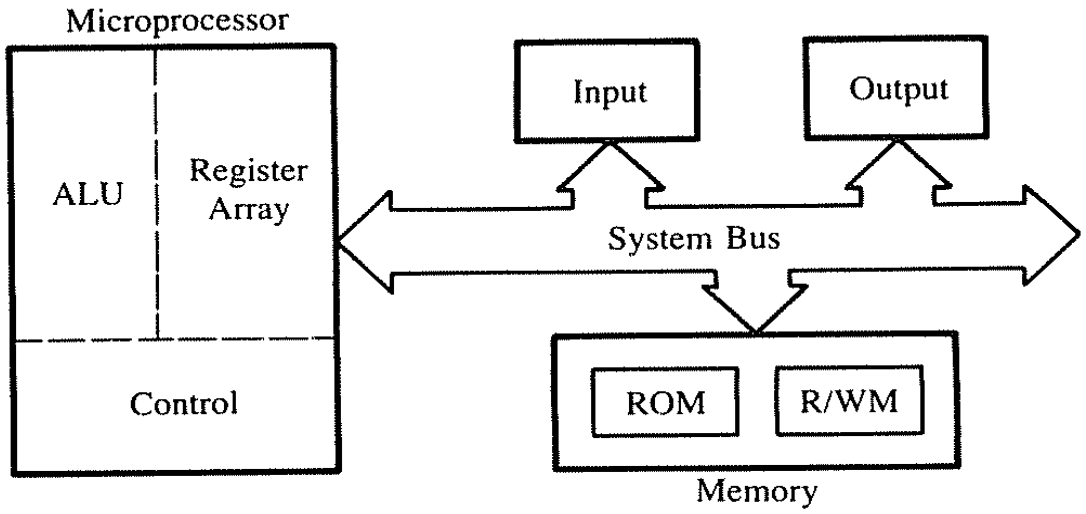


FIGURE 1.3
Microcomputer with Bus Architecture

ریزپردازنده

- ریزپزدازنده یک وسیله نیمه هادی است که شامل مدارت منطقی الکترونیکی است که با تکنیک LSI و یا VLSIساخته شده است. این وسیله شامل قسمت هایی چون واحد کنترل (CU) ، واحد ریاضی و منطقی (ALU) و تعدادی ثبات می باشد

- ریزپردازنده قادر به انجام عملیات مختلف محاسباتی از طریق اجرای برنامه و قابلیت تصمیم گیری در خصوص تغییر توالی اجرای برنامه نیز می باشد.

واحد ریاضی و منطق(ALU):

- این قسمت جایی است که عملیات مختلف محاسباتی و منطقی برروی داده ها انجام میگیرد. ALU عملیات ریاضی مانند جمع و تفریق و عملیات منطقـی ماننـد AND ، OR و XOR را انجام داده و نتیجه را در ثبات یا حافظه ذخیره می نماید.

ثبات ها:

- ثبات ها اساسا برای ذخیره نمودن داده ها بصورت موقتی در حین اجرای برنامه مورد استفاده قرار می گیرند. بعضی از این ثبات ها از طریق دستور العمل ها قابل دسترس کاربر می باشند.

واحد کنترل

- واحد کنترل سیگنال های کنترلی و زمانبندی لازم را برای کلیه عملیاتی که در یک میکروکامپیوتر صورت می گیرد فراهم می نماید. این واحد جریان داده ایی بین ریزپردازنده و حافظه و وسایل ورودی و خروجی را مهیا می سازد.

- وقتی که ریزپردازنده جهت اجرای یک برنامه دستورات را یکی یکی از حافظه می خواند، ریزبرنامه های مناسب از طریق واحد کنترل به اجرا در میاید که در نهایت منجر به انجام کاری می شود که آن دستورالعمل می بایست انجام دهد.

ورودی

- ورودی دستورالعمل ها و داده هـا را بصـورت بـاینری از دنیـای خـارج بـه ریزپردازنده منتقل می نماید.

خروجی

- قسمت خروجی، داده ها را از ریزپردازنده به وسایل خروجی مانند دیود های نوری ، مانیتور و یا چاپگر منتقل می کند.

حافظه

- حافظه اطلاعات باینری مانند داده ها و دستورالعمل ها را ذخیره و هر زمان که ریزپردازنده به آنها نیاز داشته باشد در اختیار ریزپردازنده قرار می دهد. برای اجرای برنامه، ریزپردازنده دستورات و داده ها را ازحافظه می خواند و عملیات محاسباتی را در ALU انجام میدهد. نتایج یا به وسایل خروجی جهت نمایش ارسال می شوند یا در حافظه ذخیره می گردند که بعدا مورد استفاده قرار گیرند.

گذرگاه سیستم

- گذرگاه سیستم یک مسیر ارتباطی بین ریزپردازنده و وسایل جانبی می باشد. این گذرگاه چیزی جز تعدادی سیم نیست که بیت ها را جابجا می کند.

- کلیه وسایل جانبی و حافظه ها از گذرگاه مشابه استفاده می کنند. اگر چه ریزپردازنده فقط با یک وسیله در هر مقطع زمانی ارتباط بر قرا می کند، مدیریت زمان استفاده از گذرگاه به عهده واحد کنترل ریزپردزانده می باشد.

میکروکامپیوتر چگونه کار می کند؟

- فرض کنید برنامه از قبل درون حافظه RAM ذخیره گردیده است. وقتی به میکروکامپیوتر فرمان اجرای برنامه داده میشود، میکرو دستورات را یکی یکی خوانده و اجرا می نماید و در نهایت نتیجه را به خروجی برای نمایش ارسال می کند.

- توالی اجرای برنامه فراخوانی (fetch)، رمزگشایی(decode) و اجرا(execution) می باشد.

- در طی اجرای برنامه ریزپردازنده از گذرگاه سیستم برای فراخوانی دستورات و داده ها و از ثبات ها نیز برای ذخیره موقتی داده ها و از ALU برای انجام عملیات حسابی و منطقی نیز استفاده می کند.

مجموعه دستورالعمل های ریزپردازنده و زبان های کامپیوتری

- دستورالعمل به عنوان وظیفه ایی (مانند دستور جمع کردن ADD) که ریزپردازنده می تواند انجام دهد تعریف شده است و هر دستورالعمل از یک یا چند کلمه تشکیل می شود.

- هر ریزپردازنده طبق طراحی داخلی خود دارای مجموعه دستورالعمل های مخصوص به خود می باشد.

- دستورات باید به زبان بانیری(زبان ماشین)که برای ریز پردازنده قابل فهم باشند نوشته شود.

زبان ماشین

- تعداد بیتهای یک کلمه برای یک ماشین خاص ثابت بوده و کلمات مختلف از ترکیب این تعداد بیت های ثابت بوجود می آیند. برای مثال ماشینی با طول کلمه ۸ بیت می تواند $۲^۸$ یا ۲۵۶ ترکیب ۸ بیتی یا زبانی با ۲۵۶ کلمه داشته باشد.
- مهندسین طراحی ریزپردازنده ها، با کنار هم قرار دادن بیتها، رشته های ۸ بیتی را شکل داده و به هریک از این کلمات معنی خاصی را با استفاده از مدارت منطقی نسبت می دهند. به هریک از این کلمات (رشته های صفر و یک) دستورالعمل (instruction) گفته می شود.

- معانی کلمات و دستورات ماشینی هر ریزپردازنده فقط برای همان ماشین قابل فهم خواهد بود.

زبان ماشین Z80

- z80، ریزپردازنده ایی با طول کلمات ۸ بیتی و دارای ۱۵۹ دستورالعمل می باشد.
- یک دستورالعمل، رشته ایی باینری (صفر ویک) می باشد که از طریق وسیله ورودی وارد و به ریزپردازنده فرمان می دهد تا عمل خاصی را انجام دهد. برای مثال :

0011 1100	is an instruction that increments the number in the register called the accumulator by one.
1000 0000	is an instruction which adds the number in the register called B to the number in the accumulator, and keeps the sum in the accumulator.

دستور باینری 0011 1100 به صورت 3C نوشته می شود. با فشردن کلید 3 و کلید C این دستور وارد می شود. برنامه مبصر(monitor program) کد هگزا دسیمال وارد شده را به رشته باینری تبدیل می نماید.

زبان اسمبلی Z80

- اگرچه می توان دستورات را به صورت کد های هگزا دسیمال نوشت، ولی درک چنین دستوراتی همواره مشکل می باشد.

از این رو هر سازنده ریزپردازنده برای هریک از دستورات خود کدهای حرفی (استفاده از حروف الفبا) را معرفی که به آن Mnemonic گفته می شود.

کد های حرفی با استفاده از چند حروف الفبا کاری را که دستور باید انجام دهد را مشخص می سازد.

برای مثال کد باینری 0011 1100 یا کد هگزا دسیمال 3C از ریزپردازنده Z80 بصورت INC A نمایش داده می شود.

زبان اسمبلی Z80

- توضیح کامل هریک از دستورات باید توسط سازنده ریزپردازنده فراهم و در دسترس کاربر قرار گیرد.
- مجموعه کامل دستورات حرفی Z80 (mnemonic)، زبان اسمبلی Z80 نامیده می شود و برنامه ی نوشته شده با استفاده از این زبان را برنامه زبان اسمبلی می نامند.

برنامه زبان اسمبلی نوشته شده برای یک ریزپردازنده را نمی توان در کامپیوتری که از ریزپردازنده ی دیگری استفاده می کند اجرا نمود، مگر آنکه دو ریزپردازنده در کدهای مورد استفاده سازگار باشند.

زبان ماشین و زبان اسمبلی هر دو وابسته به ماشین بوده و به زبان های سطح پایین (low level language) معروفند.

- در یک میکروکامپیوتر دستورات اسمبلی با استفاده از کدهای ASCII و از طریق صفحه کلید وارد می شود. و تبدیل این دستورات به کد های باینری (زبان ماشین) توسط برنامه دیگری به نام اسمبلر انجام می شود.

- در میکروکامپیوتر های تک بوردی که بیشتر در آزمایشگاه مورد استفاده قرار می گیرد، کاربر خود باید با استفاده از جداول موجود هریک از Mnemonicها را تبدیل به کد هگزا دسیمال کرده و سپس از طریق صفحه کلید هگزا دسیمال آنها را وارد و درنهایت اجرا نماید.
- این عمل تبدیل Mnemonic به کد هگزا دسیمال که توسط کاربر انجام می شود را اسمبل دستی (hand assembly)می نامند.

واحد ریزپردازنده عمومی (Generalized MPU)

- ریزپردازنده یک وسیله برنامه پذیر منطقی است که با یک مجموعه ایی ازدستورالعمل ها طراحی گردیده است.
- در طول اجرای یک برنامه، ریزپردازنده دائما با حافظه و وسایل ورودی/خروجی در ارتباط بوده و فرایند فراخوانی، رمزگشایی و اجرا را تکرار می نماید. سوالی که پیش می آید این است که آیا ریزپردازنده می تواند در حین اجرای برنامه به درخواست یا یک اتفاق ناگهانی دیگر پاسخ دهد؟ یا مورد وقفه قرا گیرد؟

- پاسخ به همه سوالات فوق مثبت می باشد. ریزپردازنده علاوه بر پردازش داده ها طبق برنامه موجود در حافظه باید به درخواست های مختلفی که در بالا مطرح شده است نیز پاسخ دهد

وسایل خارجی (جانبی) باید قادر به متوقف نمودن ریزپردازنده جهت جلب آن به درخواست های خود باشند.

این فرایند ارتباطی و عملیات مربوطه بین ریزپردازنده و دیگر وسایل خارجی (حافظه و I/O) را می توان در دو گروه زیر طبقه بندی نمود:

۱- عملیاتی که توسط برنامه آغاز می گردد
۲- عملیاتی که توسط وسایل خارجی(جانبی) آغاز می گردد.

عملیاتی که توسط برنامه آعاز می گردد:

برای ارتباط با حافظه و I/O، MPU باید چهار عمل را انجام دهد:

۱- خواندن حافظه (memory Read): خواندن دستورات و داده ها از حافظه
۲- نوشتن در حافظه (Memory Write) : نوشتن دستورات و داده ها در حافظه
۳- خواندن ورودی(I/O Read) : دریافت داده از وسایل ورودی
۴- نوشتن خروجی(I/O Write) : ارسال داده ها به وسایل خروجی

سوال این است که : MPU چگونه محل حافظه یا وسیله ورودی /خروجی را پیدا می کند؟

MPU هریک از ثبات های حافظه (محل های حافظه) و ثبات های ورودیی/خروجی را با استفاده از اعداد باینری که به آن **آدرس** می گویند، شناسایی می کند.

سوال بعدی این است که: چگونه MPU وسایل جانبی را برای خواندن و یا نوشتن مطلع می کند؟

این کار را با ارسال سیگنال های زمانبندی(سیگنال های کنترلی) قبل از انتقال داده ها انجام میدهد.

با توجه به نکات مطرح شده عملیاتی که MPU باید انجام دهد را می توان به صورت زیر خلاصه نمود:

۱- تشخیص محل حافظه یا وسیله ورودی/خروجی از طریق آدرس آن
۲- فراهم نمودن سیگنال های کنترلی یا هماهنگی
۳- انتقال داده های باینری

بنابراین، MPU به سه مجموعه از خطوط ارتباطی که به آنها گذرگاه (BUS) گفته می شود نیاز مند است:

۱- گذرگاه آدرس (Address bus) : برای پیدا نمودن محل های حافظه

۲- گذرگاه داده (Data bus) : برای جابجایی داده ها

۳- گذرگاه کنترل (Control bus): برای سیگنال های زمانبندی یا هماهنگی

گذرگاه آدرس (Address bus)

- MPU از طریق آدرس باینری محل های حافظه یا وسایل جانبی را پیدا میکند.

سوالی که مطرح می شود این است که: اندازه این آدرس چقدر است؟ یا بزرگی این آدرس چقدر است؟ یعنی از چند بیت تشکیل شده است. جواب این سوال بستگی به طراحی داخلی ریزپردازنده و تعداد پایه های موجود بر روی تراشه دارد که میتواند ۸، ۱۶، ۲۰ یا بیشتر باشد.

این گذرگاه یک طرفه است. سیگنال ها از طرف MPU به وسایل جانبی روانه می شوند زیرا MPU سیگنا ل های آدرس را تولید و ارسال می نماید.

خطوط آدرس معمولا به صورت A_0 تا A_m نشان داده می شوند

گذرگاه داده (Data bus)

- گروه دیگری از خطوط که در شکل 2.1نشان داده شده است، گذرگاه داده می باشد. از این خطوط که به صورت دو طرفه نیز می باشند برای انتقال داده ها استفاده می شود. این خطوط به صورت D_0 تا D_n نشان داده شده اند. اندازه این گذرگاه از نظر تعداد سیم مشخص میکند که چه تعداد بیت را می توان در هر مقطع زمانی رد و بدل نمود و اندازه این گذرگاه در معماری ریزپردازنده تاثیر بسیار زیادی دارد. ریزپردازنده های Z80، 8080و 6800هرکدام دارای گذرگاه داده ی ۸ بیتی می باشند و به آنها ریزپردزانده های ۸ بیتی نیز می گویند.

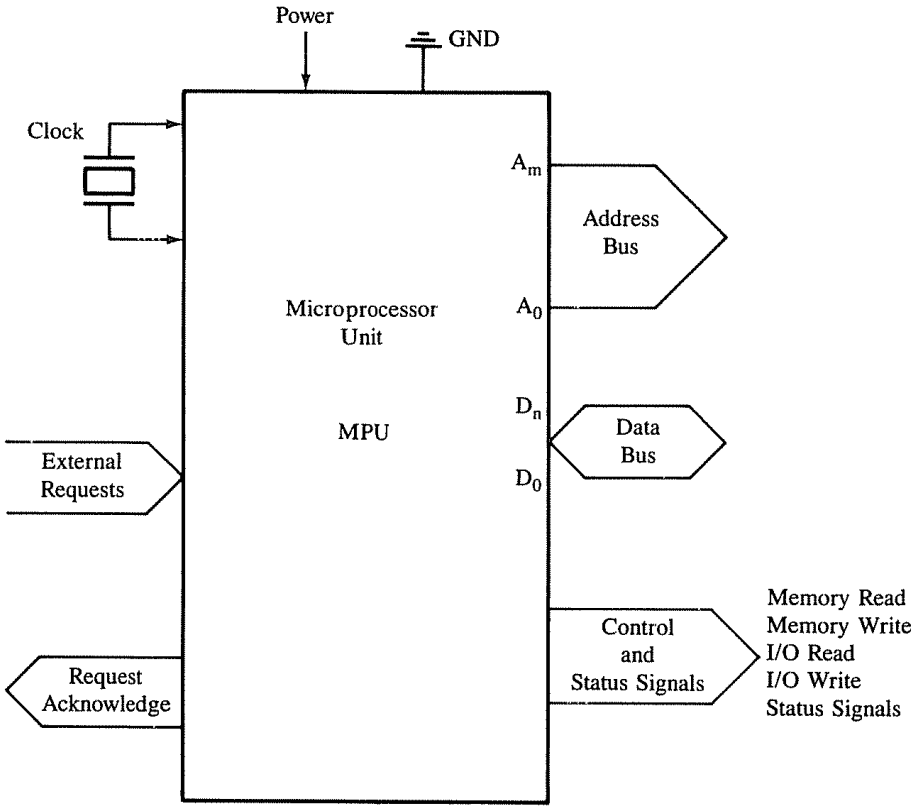


FIGURE 2.1
Generalized Microprocessor Unit (MPU)

خطوط کنترل

- این خطوط تعدادی خطوط انفرادی هستند که توسط MPU تولید تا عملیاتی را که میخواهد انجام دهد مشخص سازد. MPU برای هریک از چهار عملی که میخواهد انجام دهد یک سیگنال کنترلی مجزا تولید و ارسال می نماید. این چهار سیگنال عبارتند از:

Memory read، Memory write، I/O Read و I/O write

این سیگنال ها سیگنال های زمانبندی هستند که جهت فعال نمودن وسایل جانبی بکار میروند.

برای مثال برای فراخوانی یک دستورالعمل از حافظه، MPU سیگنال Memory readرا جهت فعال سازی تراشه حافظه ارسال می کند.

عملیاتی که توسط وسایل جانبی آغاز می شوند

مواقعی وجود دارد که عملیات در حال اجرا توسط MPU نیاز به وقفه دارد. وقفه های خارجی که باعث توقف MPU مورد بحث ما می شود را می توان به چهار گروه تقسیم بندی نمود:

۱- Reset: شروع دوباره از ابتدا.

۲- Interrupt: توقف برنامه در حال اجرا به صورت موقتی و باز گشتن به برنامه بعد از اتمام سرویس

۳- Wait: وقتی پاسخ زمانی حافظه بسیار کندتر از MPU میباشد، این سیگنال استفاده میشود تا عملیات MPU را به تاخیر بیاندازد.

۴- Bus Request: وقتی عملکرد MPU نسبت به وسایل جانبی کندتر باشد. وسایل جانبی درخواست استفاده از گذرگاه را صادر می کند.

در شکل 2.1 سیگنال های فوق در قالب خطوط درخواست های خارجی (External Request) نشان داده شده است.

MPU برای پاسخ به درخواست های خارجی نیاز به تعدادی خطوط اضافی تحت عنوان تایید درخواست (Request Acknowledge) دارد که در شکل 2.1 نشان داده شده است.

سیگنال CLOCK و POWER

- MPUرا می توان به عنوان یک زمان سنج (Counter) پیچیده در نظر گرفت. زمان سنجی یا کنترل و مدیریت زمان انجام کار در عملیات مختلفی که MPU انجام میدهد بسیار حیاتی می باشد. بیت های یک دستورالعمل باینری با ریزبرنامه های درون تراشه در ارتباط می باشد. وقتی MPUدستورالعملی را اجرا می کند، یک سری از ریزبرنامه ها توسط MPUدر بازه های زمانی منظم آزاد می شوند. بنابراین، MPU به مداراتی که سیگنال های ساعت را تولید می کنند نیاز مند می باشد. تراشه MPU همچنین نیاز به پاور جهت انجام کلیه کارهای خود دارد.

ریزپردازنده به عنوان واحد پردازش

- ریزپردازنده اجرای دستورالعمل ها را در یک فرآیند پشت سر هم و پیوسته که شامل فراخوانی ، رمزگشایی و اجرا می باشد انجام میدهد.

فراخوانی یک دستورالعمل

برای فراخوانی یک دستورالعمل، ریزپردازنده آدرس محل حافظه مورد نظر را برروی گذرگاه آدرس قرار داده و اطلاعات باینری را از طریق گذرگاه داده دریافت می کند (می خواند). بنابراین ریزپردازنده به ثباتی نیاز دارد تا بتواند آدرس ها ی حافظه را درخود جای دهد و قابلیت افزایش یکی به آدرس درون خود را نیز داشته باشد.

رمزگشایی دستورالعمل

وقتی یک دستورالعمل فراخوانی گردید، نیاز به رمزگشایی دارد تا بتواند به سوالات زیر پاسخ دهد.

۱- آیا این یک دستور کامل است؟ اگرنه چند بایت دیگر باید فراخوانی گردد؟

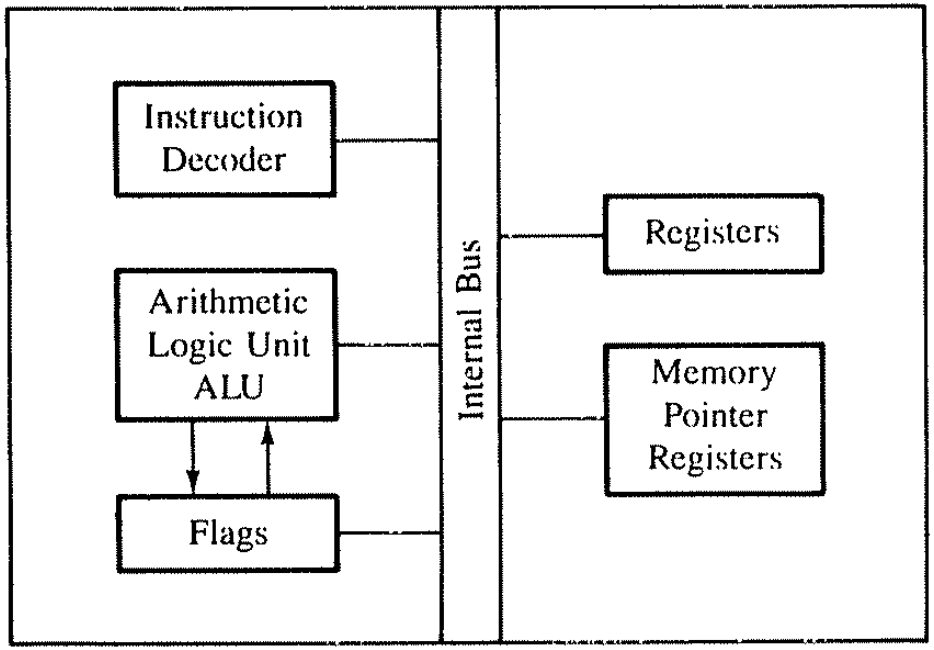
۲- چه نوع عملی باید انجام شود؟ و بروی چه داده ایی باید انجام شود؟

برای انجام این عملیات ریزپردازنده نیاز به یک رمزگشای دستورالعمل (Instruction decoder) دارد تا بتواند دستور باینری فراخوانی شده را تفسیر نماید.

اجرای یک دستورالعمل

- نوع عملی که یک ریزپردازنده می تواند انجام دهد بستگی به ریزبرنامه هایی دارد که در داخل آن قرار داده شده است. به عبارت دیگر بستگی به مجموعه دستورالعمل های ریزپردازنده دارد.
- عملیاتی که ریزپردازنده می تواند انجام دهد در انواع مختلفی طبقه بندی می گردد. مانند، انتقال داده ها، عملیات ریاضی و منطقی و عملیات تصمیم گیری و .....
- نیازمندی های یک ریزپردازنده برای پردازش داده ها را می توان در شکل 2.2 نشان داد. از روی این بلوک دیاگرام می توان مدل برنامه نویسی یک ریزپردازنده خاص را بدست آورد.

FIGURE 2.2
MPU Internal Structure



بررسی مجدد مفاهیم مهم:

نیاز های اساسی یک ریزپردازنده عمومی در شکل 2.3 نشان داده شده است:

برای ارتباط با حافظه و وسایل ورودی خروجی MPU باید دارای **امکانات زیر باشد:**

۱- گذرگاه آدرس، برای ارسال آدرس یک ثبات حافظه یا I/O
۲- گذرگاه داده، برای انتقال داده بین MPU و حافظه و وسایل ورودی خروجی
۳- سیگنال های کنترلی، برای تشخیص و تعیین عملیات و فراهم نمودن زمانبندی مناسب.
۴- خطوط سیگنال های درخواست خارجی، برای متوقف نمودن عملیات جاری MPU
۵- سیگنال های تایید درخواست، برای پاسخ به در خواست های وسایل جانبی
۶- سیگنال ساعت برای فراهم نمودن زمانبندی و پاور برای فعال نمودن مدارها

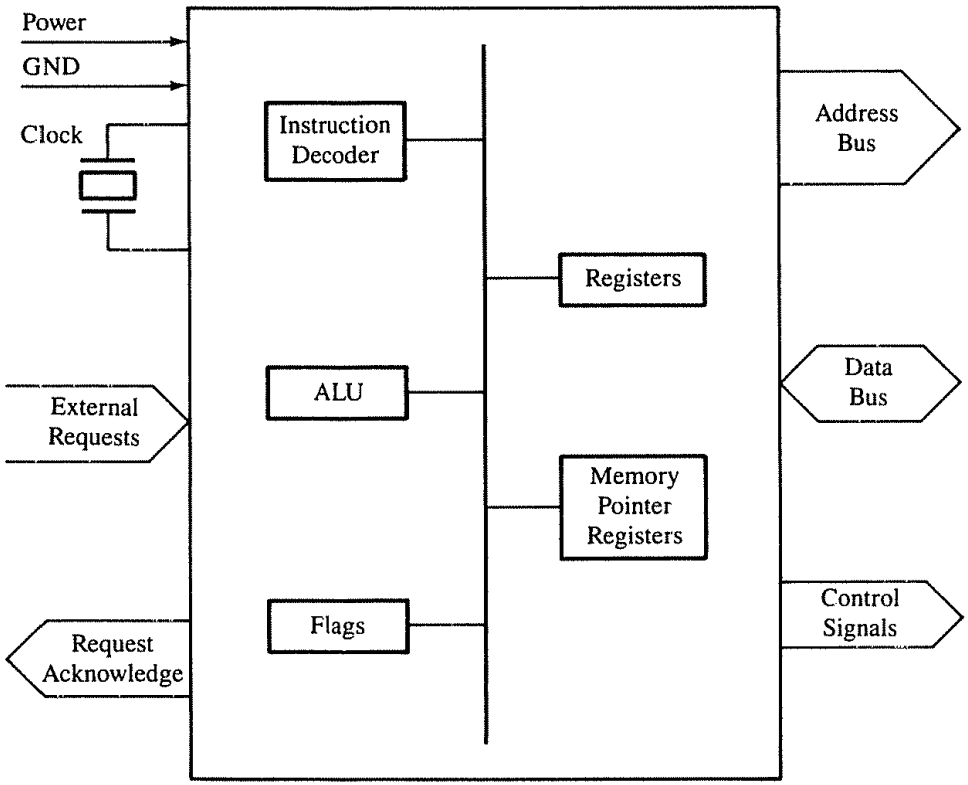


FIGURE 2.3
MPU Architecture

برای پردازش داده بصورت داخلی ، MPU باید دارای امکانات زیر باشد:

۱- دیکودر دستورالعمل (Instruction decoder)، برای رمزگشایی نمودن اطلاعات باینری وارد شده

۲- ثبات ها، برای ذخیره نمودن داده ها

۳- ثبات هایی به عنوان اشاره گر حافظه، برای آدرس نمودن ثبات های حافظه

۴- ALU، برای انجام عملیات ریاضی و منطقی

۵- پرچم ها (Flags) ، فلیپ فلاپ هایی برای مشخص نمودن وضعیت داده ها برای تصمیم گیری

حافظه

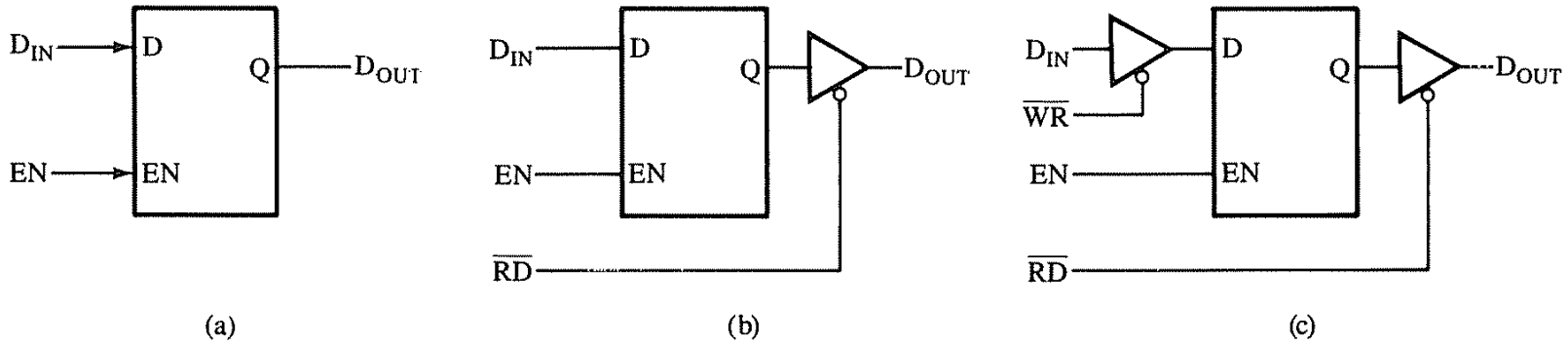
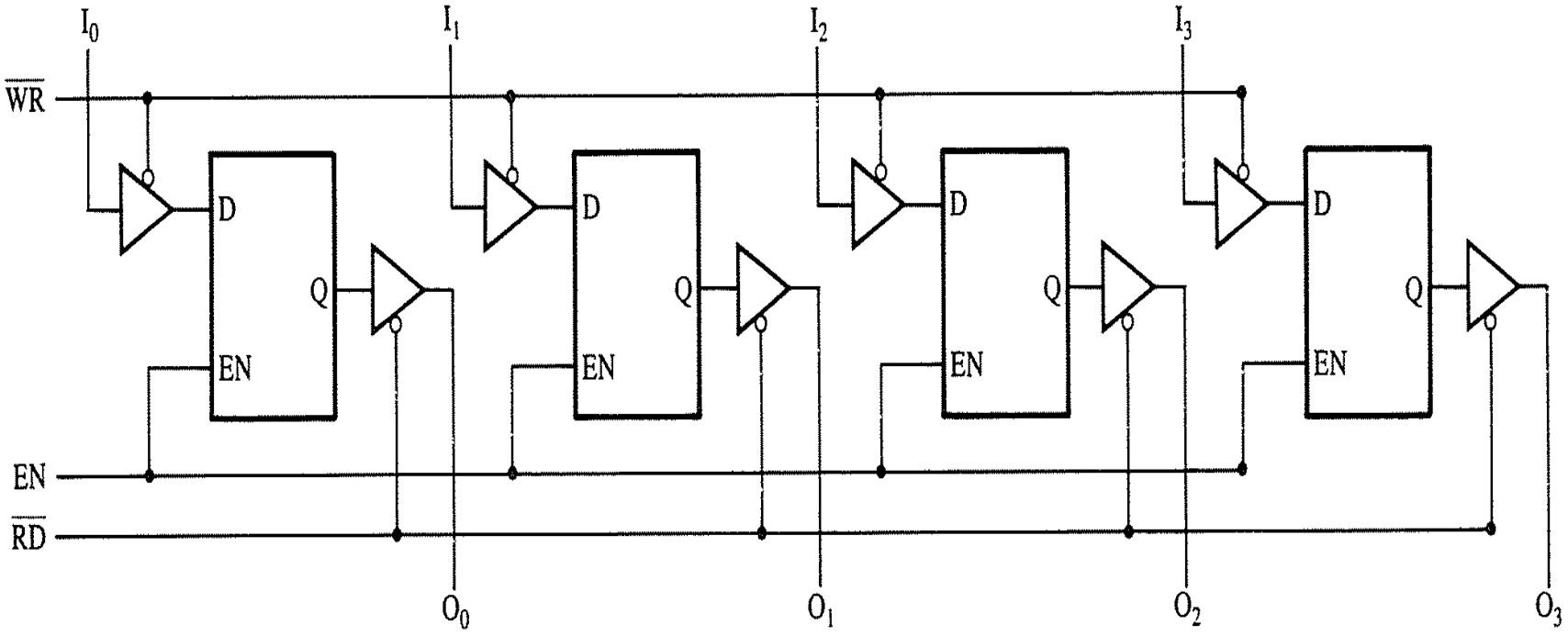
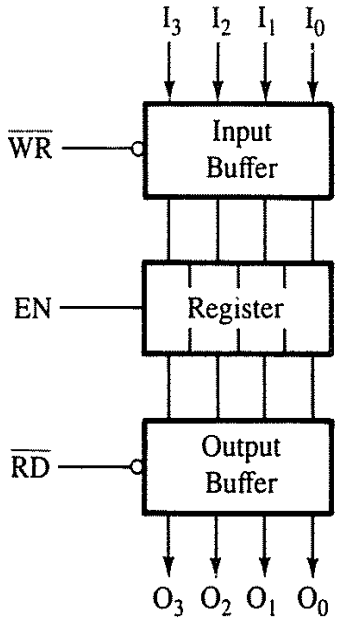


FIGURE 2.4
Latches as Storage Elements

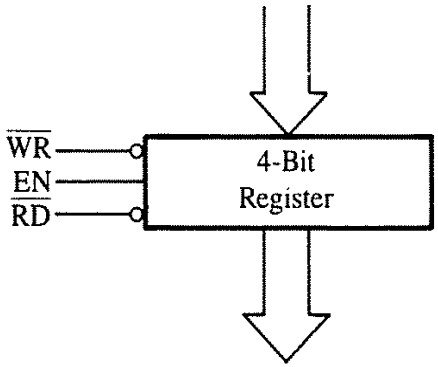


(a)

(a)



(b)



(c)

FIGURE 2.5
(a) Four Latches as a 4-Bit Register (b) and (c) Block Diagrams of a 4-Bit Register

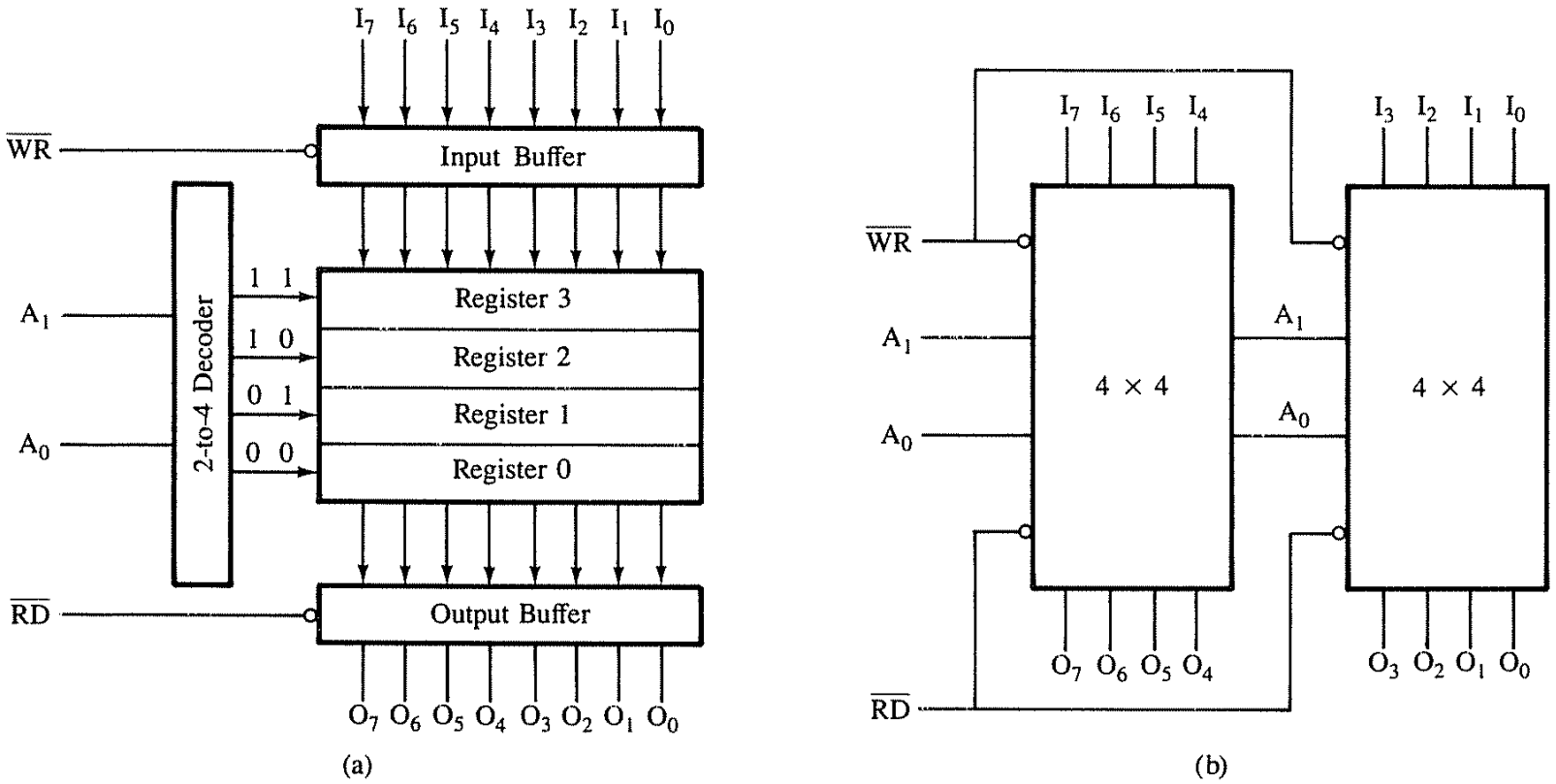


FIGURE 2.6
(a) 4 × 8 Bit Register (b) Two 4 × 4 Bit Registers

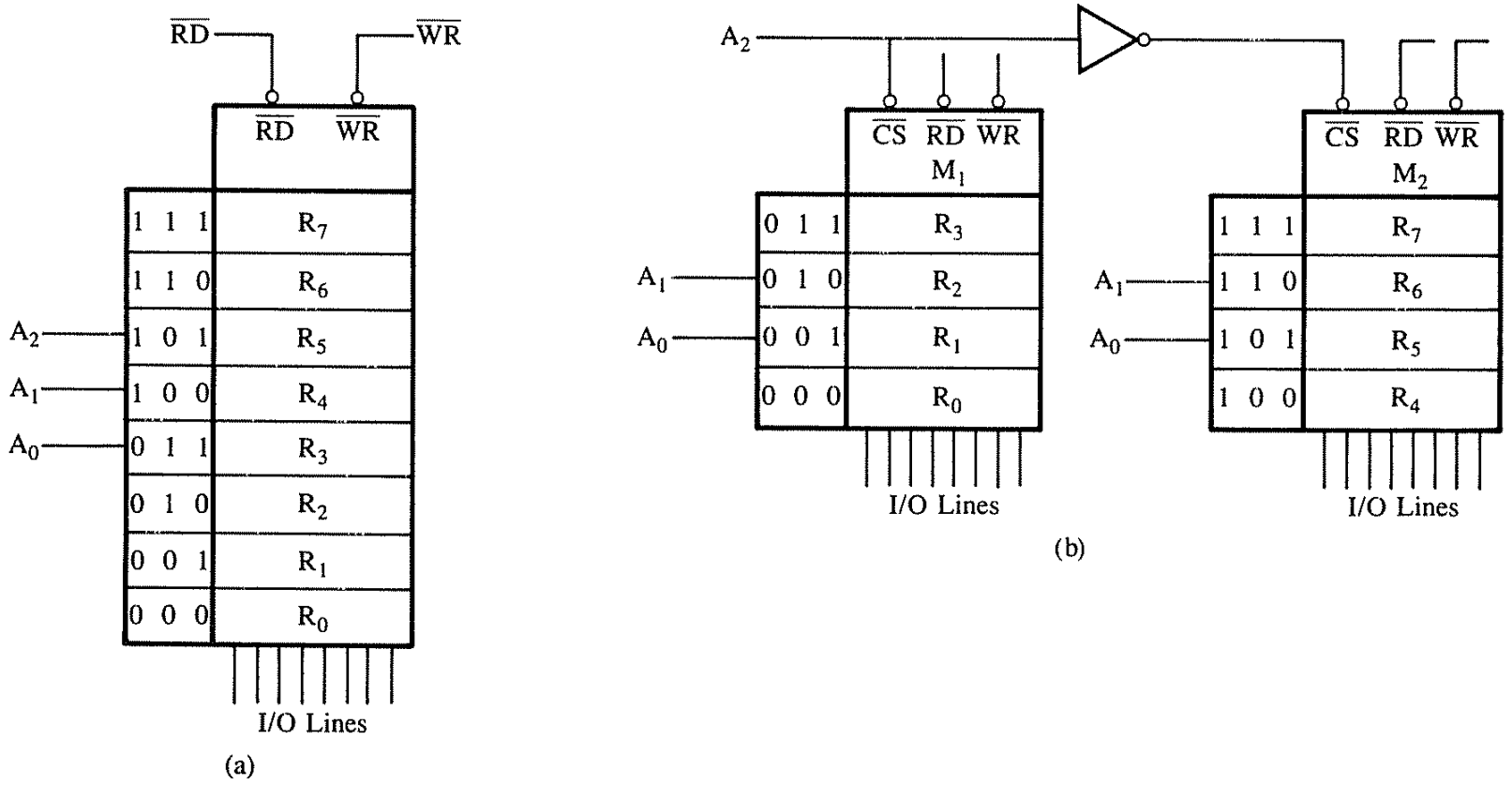


FIGURE 2.7
(a) Memory Chip with Eight Registers (b) Two Memory Chips with Four Registers Each

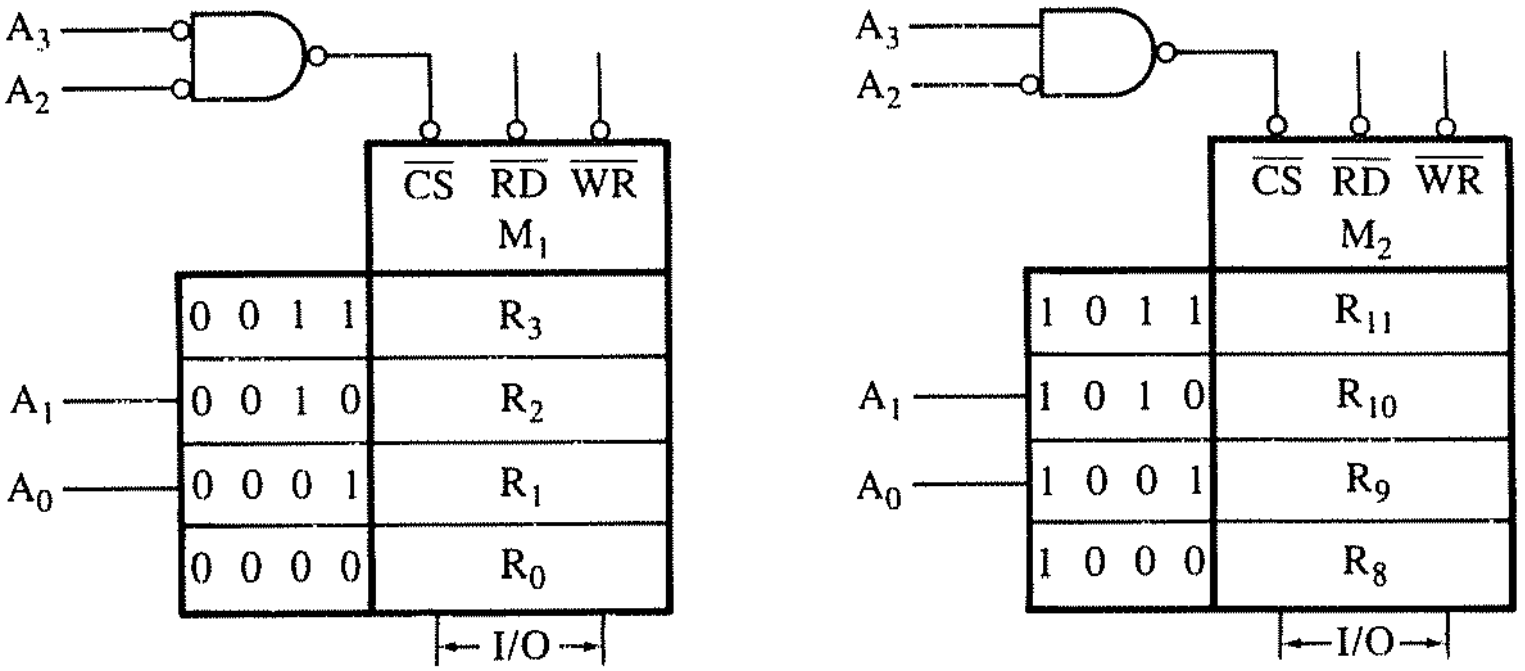


FIGURE 2.8
Addressing Eight Registers with Four Address Lines

After reviewing the above explanation, we can summarize the requirements of a memory chip as follows:

1. A memory chip requires address lines to identify a memory register, a Chip Select $\overline{CS}$ signal to enable the chip, and control signals to read from and write into memory registers.
2. The number of address lines required is determined by the number of registers in a chip (2^n = Number of registers where n is the number of address lines).
3. If additional address lines are available in a system, they are used to enable the Chip Select $\overline{CS}$ signal. The memory address of a register is determined by the logic levels (0/1) of all the address lines (including the address lines used for $\overline{CS}$).
4. The control signal Read ($\overline{RD}$) enables the output buffer, and data from the selected register are made available on the output lines. Similarly, the control signal Write ($\overline{WR}$) enables the input buffer, and data on the input lines are written into memory cells.

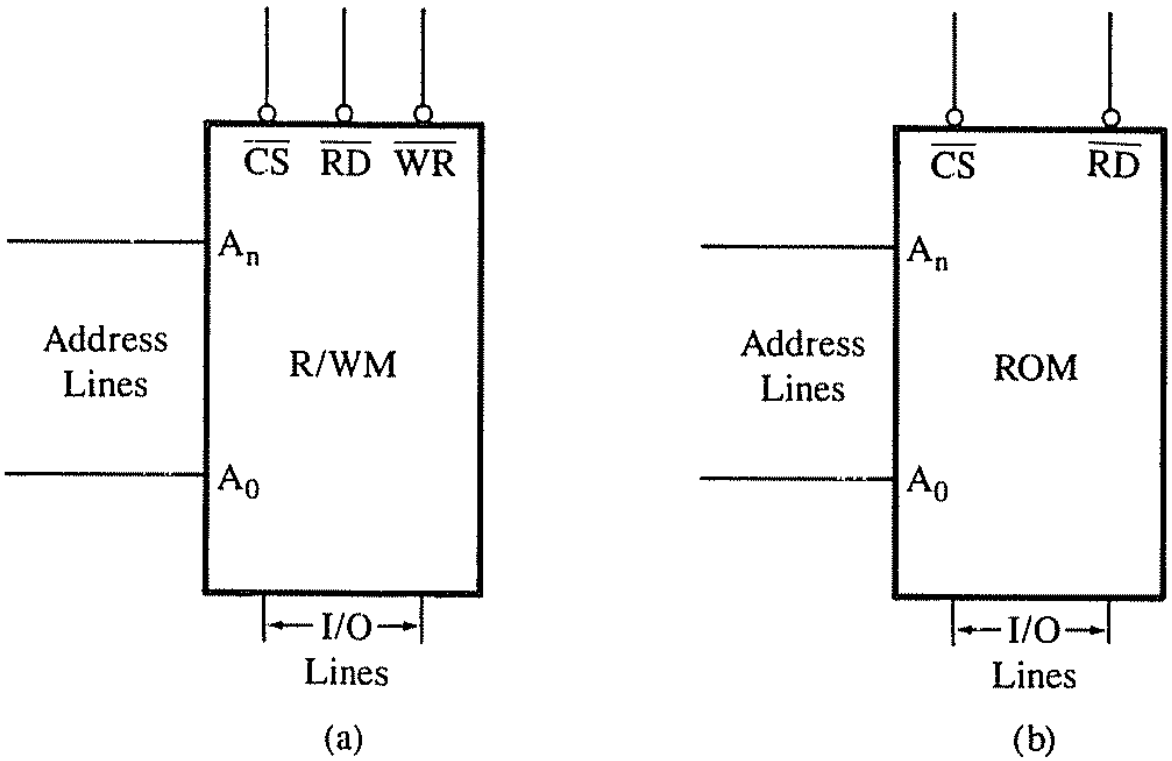


FIGURE 2.9
(a) R/W Memory Model (b) ROM Model

نقشه حافظه (Memory Map)

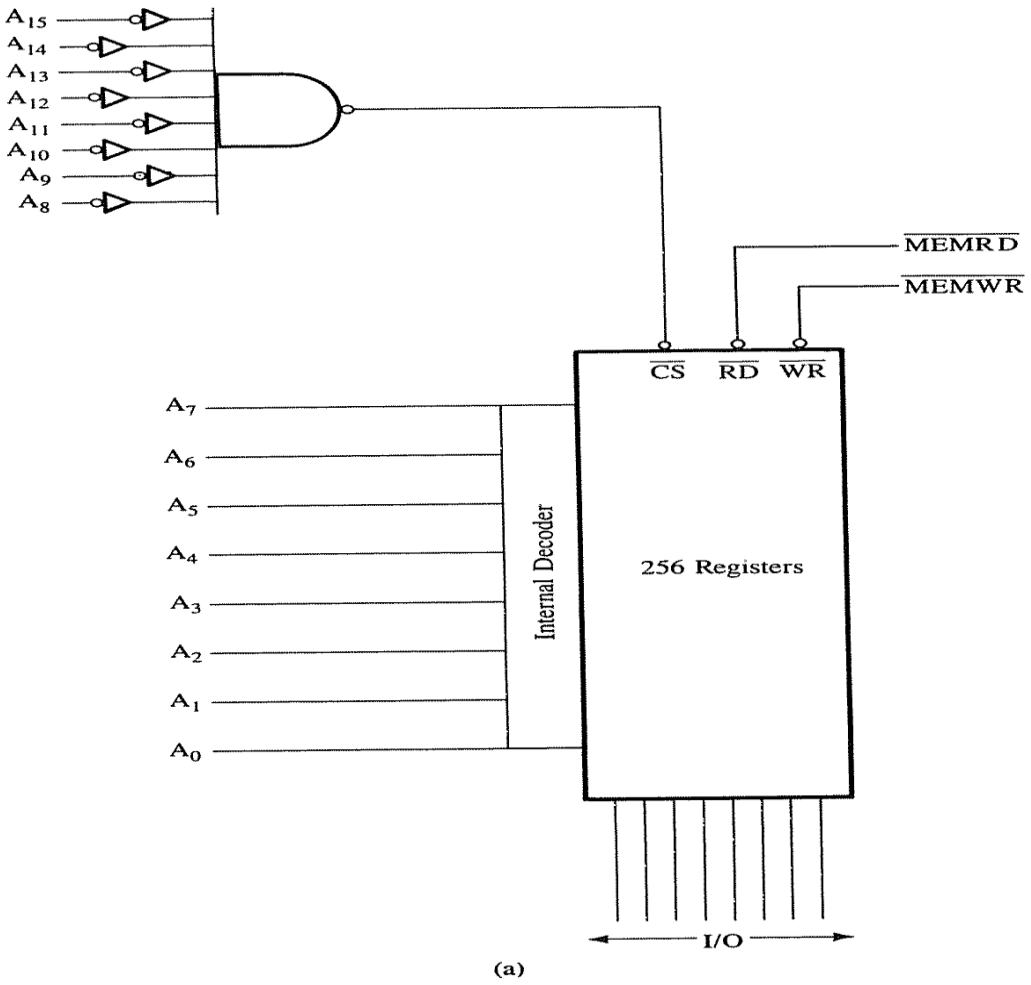
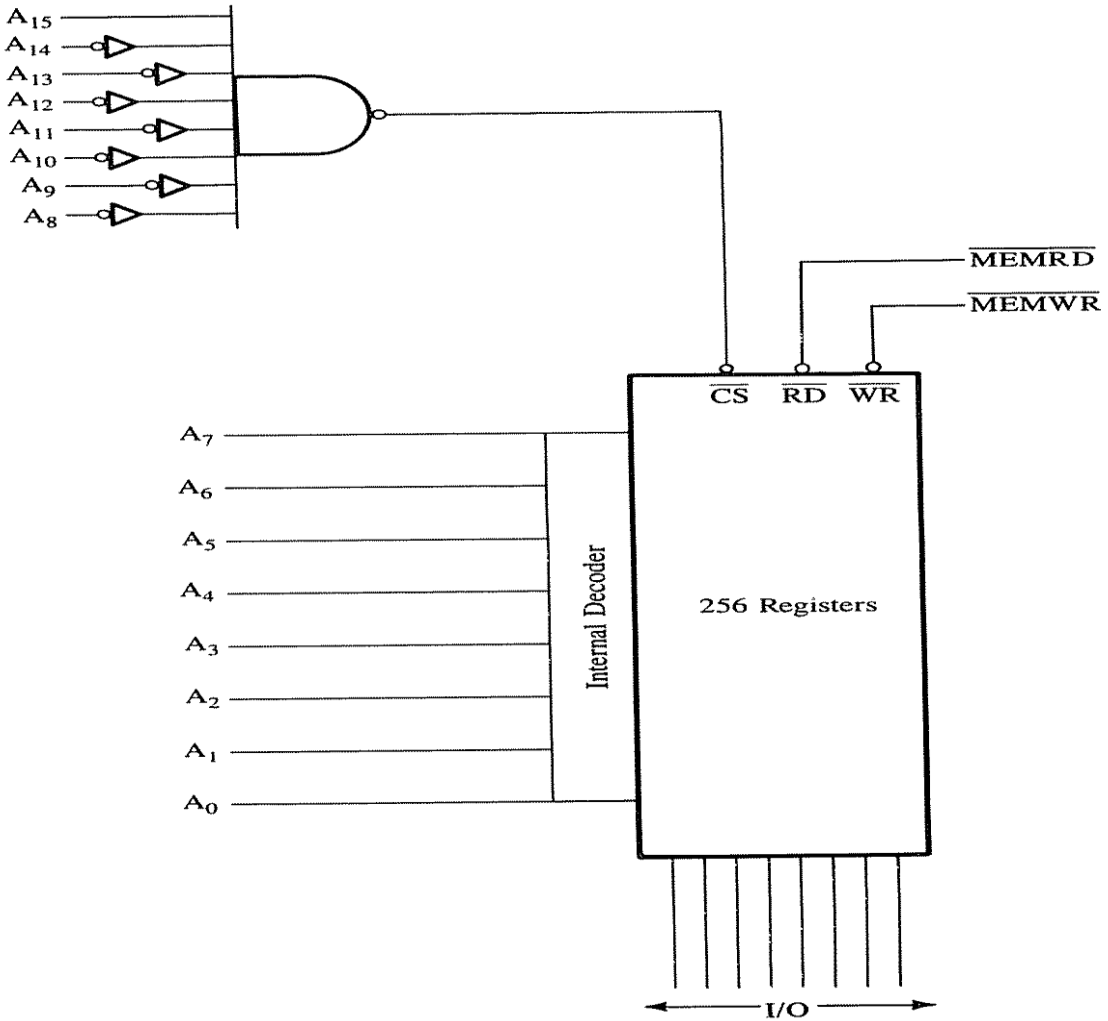


FIGURE 2.10
Memory Maps: 256 Bytes of Memory

2.10(a) will range from 0000_H to $00FF_H$ as shown below.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$= 0000_H$
								↓							↓	
								1	1	1	1	1	1	1	1	$= 00FF_H$
Chip Enable or Chip Select								Register Select								


A15
A14
A13
A12
A11
A10
A9
A8
MEMRD
MEMWR
CS
RD
WR
A7
A6
A5
A4
A3
A2
A1
A0
Internal Decoder
256 Registers
I/O

(b)

The entire range of the memory addresses from 0000_H to $00FF_H$ is known as the memory map of the chip in Figure 2.10(a). The Chip Select addresses are determined by the hardware (the inverters and NAND gate); therefore, the memory map of the chip can be changed by modifying the hardware. For example, if the inverter on line A_{15} is removed as shown in Figure 2.10(b), the address required on A_{15}–A_8 to enable the chip will be as follows:

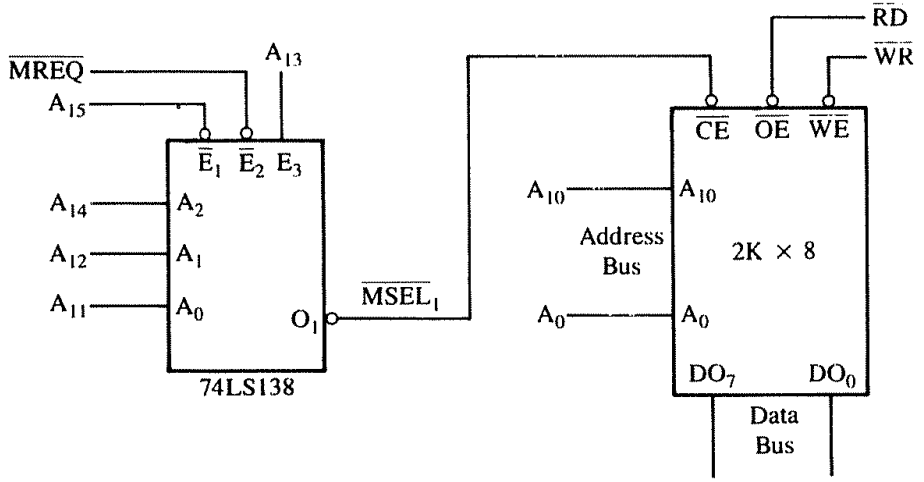
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
1	0	0	0	0	0	0	0	$= 80_H$

The memory map for Figure 2.10(b) will be 8000_H to $80FF_H$.

The memory chips in Figures 2.10(a) and (b) are the same chips. However, by changing the hardware of the Chip Select logic, the location of the memory in the map can be changed, and memory can be assigned addresses in various locations over the entire range of 0000 to $FFFF_H$.

مثال:

Given a 2K R/W (2048 × 8) static memory chip and one 3-to-8 decoder, design memory for the beginning address 2800_H. Use the $\overline{MREQ}$ signal to enable one of the decoder lines, and the $\overline{RD}$ and $\overline{WR}$ control signals can be directly connected to the memory chip.



A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	$= 2800_H$

$\overline{MSEL_1}$

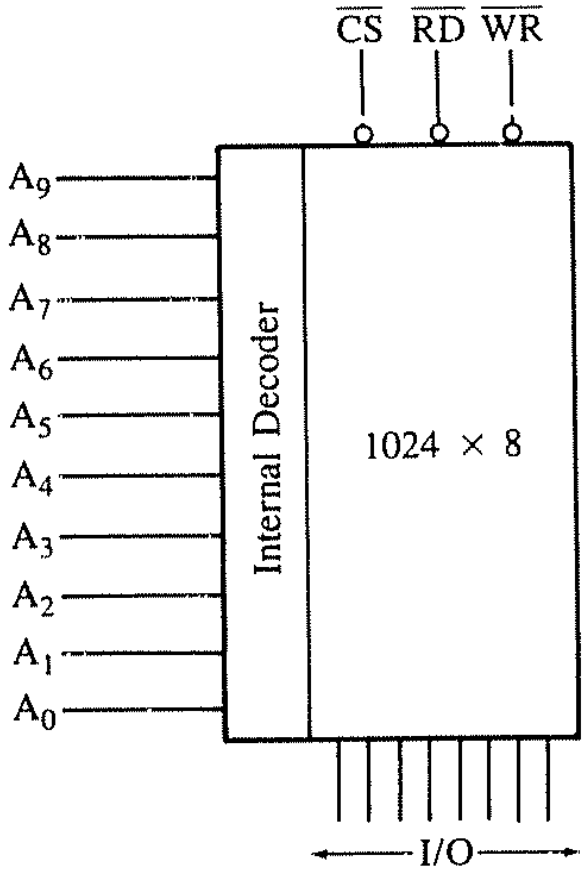
اتصال به حافظه (interfacing memory)

- مادامی که یک برنامه اجرا می شود، ریزپردازنده باید به حافظه دسترسی داشته باشد تا بتواند دستورات و داده های ذخیره شده در آن را بخواند. مدار ارتباط دهی این دسترسی را میسر می سازد. حافظه برای خواندن و نوشتن در محل های حافظه خود به تعدادی سیگنال نیاز دارد. مشابها" ریزپردازنده نیز خود تعدادی سیگنال برای برقراری ارتباط با حافظه تولید می کند. فرایند ارتباط دهی نیز به طراحی مدارهایی برای تطبیق دادن نیاز مندی های حافظه و سیگنال های تولید شده توسط ریزپردازنده اشاره دارد. در مطالب زیر به بررسی ساختار حافظه و نیاز مندی های آن و همچنین سیکل های ماشینی خواندن و نوشتن ریزپردازنده Z80 می پردازیم. سپس به قدم های لازم جهت ارتباط دهی حافظه و ریزپردازنده Z80 اشاره خواهیم کرد.

ساختار حافظه و الزامات آن

- یک حافظه خواندنی/ نوشتنی (R/WM) گروهی از ثباتها برای دخیره نمودن اطلاعات باینری می باشد. شکل 4.1 یک تراشه حافظه خواندن/ نوشتنی با ۱۰۲۴ ثبات یا محل حافظه ی ۸ بیتی (خطوط I/O) را نشان میدهد. این تراشه دارای ۱۰ خط آدرس A_0تا A_9 و یک خط انتخاب تراشه 'CS و دارای دو خط کنترلی 'RD برای فعال نمودن بافر خروجی و 'WR برای فعال نمودن بافر ورودی می باشد. شکل 4.1 همچنین یک دیکودر داخلی برای رمز گشایی خطوط آدرس دارد.

FIGURE 4.1
Logic Diagram: A Typical 1K Memory Chip



از گذشته به یاد داریم که برای خواندن از یک ثبات حافظه و نوشتن در آن، قدم های زیر باید بر داشته شود.

۱- یک آدرس باید برروی گذرگاه آدرس قرار گیرد. خطوط ارزش پایین آدرس توسط دیکودر داخلی تراشه حافظه رمزگشایی و محل حافظه مورد نظر شناسایی می گردد.

۲- خطوط آدرس ارزش بالاتر باید رمزگشایی تا سیگنال انتخاب تراشه ('CS) تولید و به تراشه مورد نظر اعمال و انتخاب شود.

۳- برای خواندن از محل حافظه آدرس شده، سیگنال 'RD (پایین گذر) باید بافر خروجی را فعال و سپس بایت مورد نظر از داخل ثبات بر روی گذرگاه داده قرار گیرد

۴- برای نوشتن در داخل ثبات آدرس شده، سیگنال 'WR (پایین گذر) باید بافر ورودی را فعال تا بایت موجود بر روی گذرگاه داده به داخل ثیات انتقال یابد.

برای ارتباط این تراشه حافظه با ریزپردازنده ، سیگنال های ریزپردازنده که باید در زمان ارتباط با حافظه اعمال گردد مورد بررسی قرار گیرد.

چگونه ریزپردازنده Z80 ازداخل حافظه می خواند یا در داخل حافظه می نویسد؟

برای خواندن از حافظه ، Z80 اعمال زیر را انجام میدهد. طبق شکل 4.2(a)

۱- یک آدرس ۱۶ بیتی را برروی خطوط گذرگاه آدرس قرار می دهد. (آدرس ها بصورت ارزش پایین و ارش بالا نشان داده شده است).

۲- سیگنال 'MREQ اعمال تا نشان دهد که گذرگاه آدرس آدرس مجاز را دارا می باشد.

۳- سیگنال 'RD را صفر نموده تا نشان دهد که می خواهد بخواند.

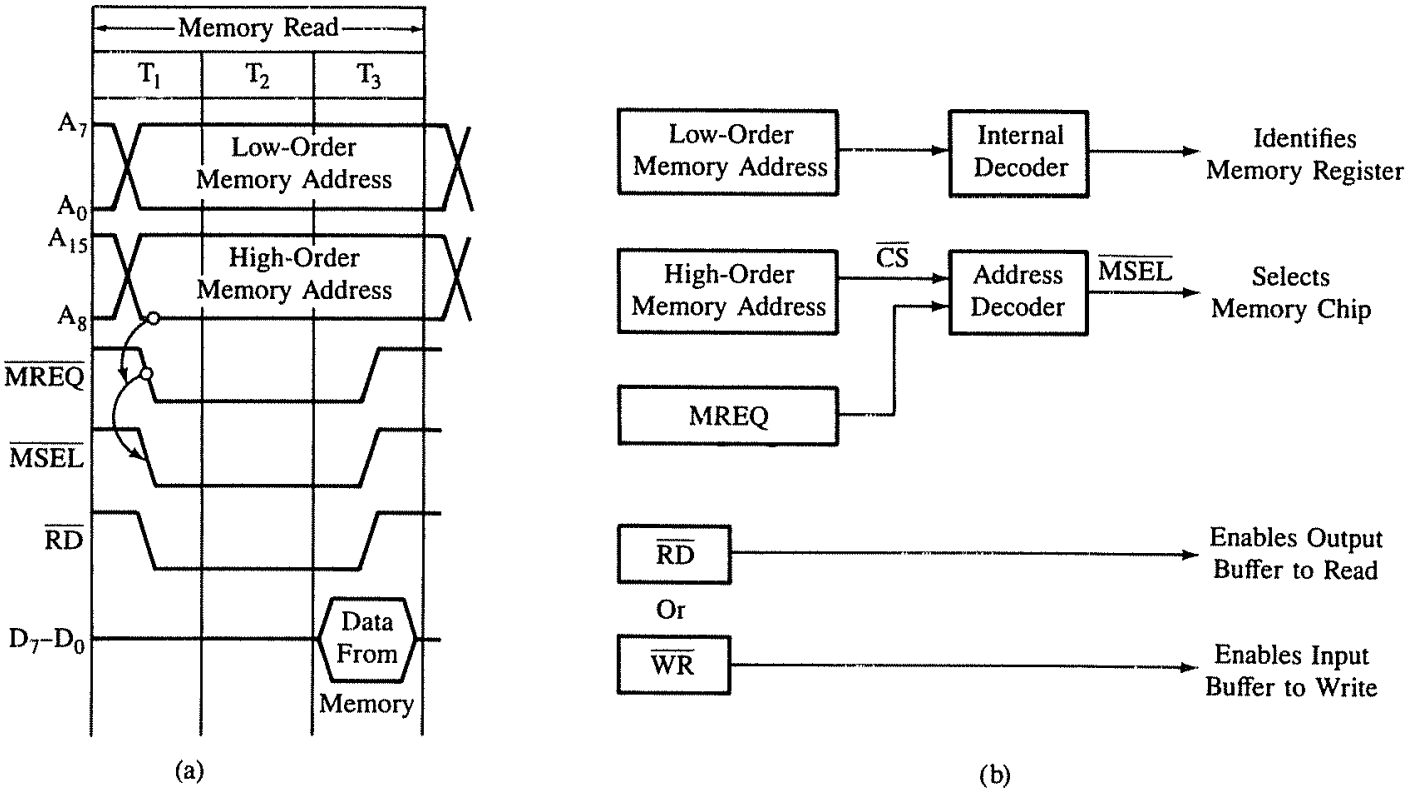
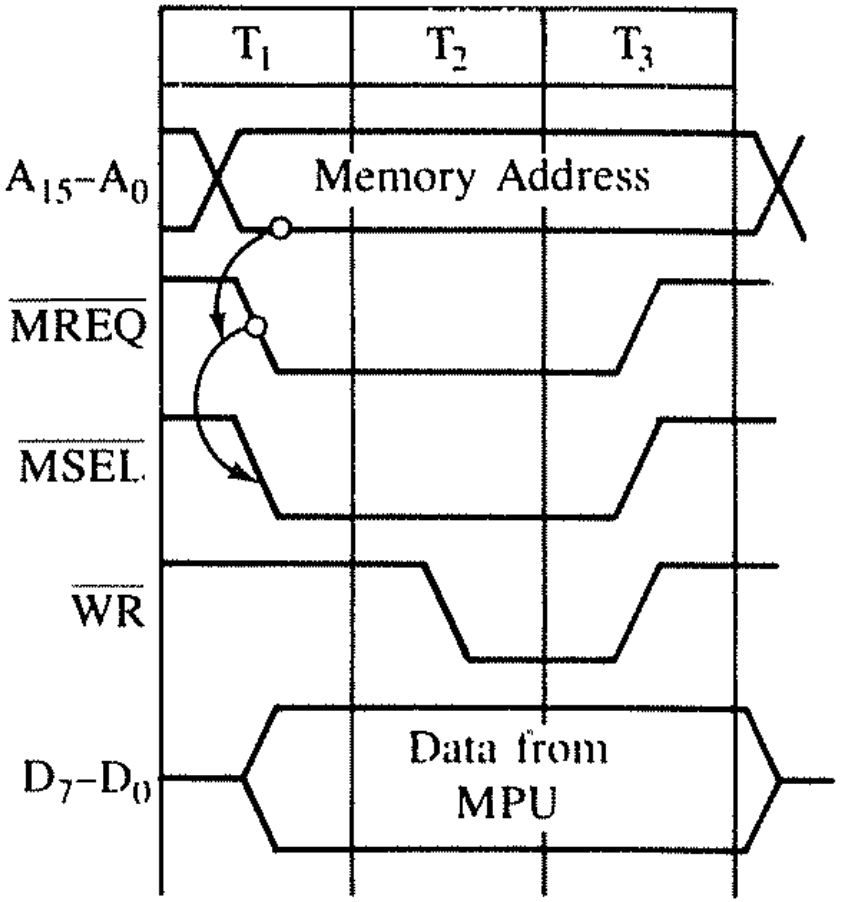


FIGURE 4.2
(a) Memory Read Timing Diagram (b) Block Diagram: Address Decoding and Memory Read/Write Operations

برای نوشتن در حافظه، Z80 اعمال زیر را انجام میدهد. طبق شکل 4.3.

۱- یک آدرس ۱۶ بیتی را برروی خطوط گذرگاه آدرس قرار می دهد.
۲- سیگنال 'MREQرا اعمال و داده ها را برروی گذرگاه داده قرار می دهد.
۳- سیگنال 'WR را اعمال میکند.

FIGURE 4.3
Writing into Memory Register



- برای درک و طراحی یک مدار ارتباط دهی ، ما باید نیازمندی های حافظه را با عملیات خواندن و نوشتن Z80 مطابقت دهیم.

مفاهیم اصلی در ارتباط دهی حافظه

عمل اصلی ارتباط دهی حافظه این است که ریزپردازنده اجازه پیدا کند که از داخل یک ثبات تراشه حافظه بخواند یا در آن بنویسد. برای انجام چنین اعمالی، ریزپردازنده باید

۱- قادر باشد تا تراشه را انتخاب کند.

۲- ثبات یا محل حافظه را شناسایی کند .

۳- بافر مورد نظر را فعال سازد.

دیگرام زمانی عمل خواندن حاافظه را بررسی نماییم. شکل 4.2(a) .

برای درک اینکه Z80 چگونه از حافظه می خواند. در شکل 4.2(a)، برای توضیح مفهوم رمزگشایی، گذرگاه آدرس به دوقسمت تقسیم گردیده است، ارزش پایین و ارش بالا.

۱- Z80 یک آدرس ۱۶ بیتی را برروی گذرگاه آدرس قرار می دهد و با آین آدرس فقط یک ثبات باید انتخاب گردد. برای تراشه حافظه شکل 4.1 فقط ۱۰ خط آدرس برای شناسایی ۱۰۲۴ ثبات نیاز می باشد. بنابراین، ما می توانیم ۱۰ خط ارزش پایین گذرگاه آدرس (A_0 تا A_9) ریزپردازنده را به تراشه حافظه متصل نماییم. دیکودر داخلی تراشه حافظه می تواند ثبات مورد نظر را شناسایی و انتخاب نماید. طبق شکل 4.2(b) .

۲- بقیه خطوط آدرس (A_{15} ـ A_{10}) Z80 باید برای تولید سیگنال انتخاب تراشه ('CS)بکار گرفته شود.

۳- ریزپردازنده Z80 دو سیگنال به نام های 'MREQ و 'RD تولید می کند. سیگنال 'MREQ می تواند با سیگنال 'CS ترکیب تا یک سیگنال به نام 'MSEL تولید کرده که از طریق آن تراشه حافظه را انتخاب می نماید.

۴- ریزپردازنده سیگنال کنترلی پایین گذر 'RD را برای فعال نمودن بافر خروجی حافظه اعمال تا داده از داخل آن خوانده شود.

شکل 4.2(a) همچنین نشان میدهد که حافظه باید داده را در شروع T_3 بر روی گذرگاه داده قرار دهد.

برای نوشتن در داخل یک ثبات، ریزپردازنده قدم های مشابه ایی را بر میدارد.

- شکل 4.3 سیکل نوشتن در حافظه را نشان میدهد. در عمل نوشـتن، Z80 آدرس و داده را برروی گذرگاه های مورد نظر قرار داده و همچنین سیگنال 'MREQ را اعمال میکند. بعد از اینکه داده برروی گذرگاه داده ثابت شد، سیگنال 'WR را اعمـال مـی کند. سیگنال 'WR بافر ورودی تراشه حافظه را فعـال تـا داده مـورد نظـر در داخـل ثبات انتخاب شده قرار گیرد.

- یک راه دیگر برای تولید سیگنال 'MSEL که تراشه حافظه را انتخاب می کند این است که سیگنال های کنترلی 'MEMRD و 'MEMWR را با ترکیب 'MREQ و 'RD و 'WR تولید نماییم. طبق شکل 4.4(a)
- سیگنال 'MEMRD را می توان برای فعال نمودن بافر خروجی حافظه جهت خواندن مورد استفاده قرا داد. و از سیگنال 'MEMWR میتوان برای فعال نمودن بافر ورودی جهت نوشتن در حافظه استفاده کرد و همچنین می توان از سیگنال 'CS برای انتخاب تراشه حافظه استفاده نمود شکل 4.4(b).

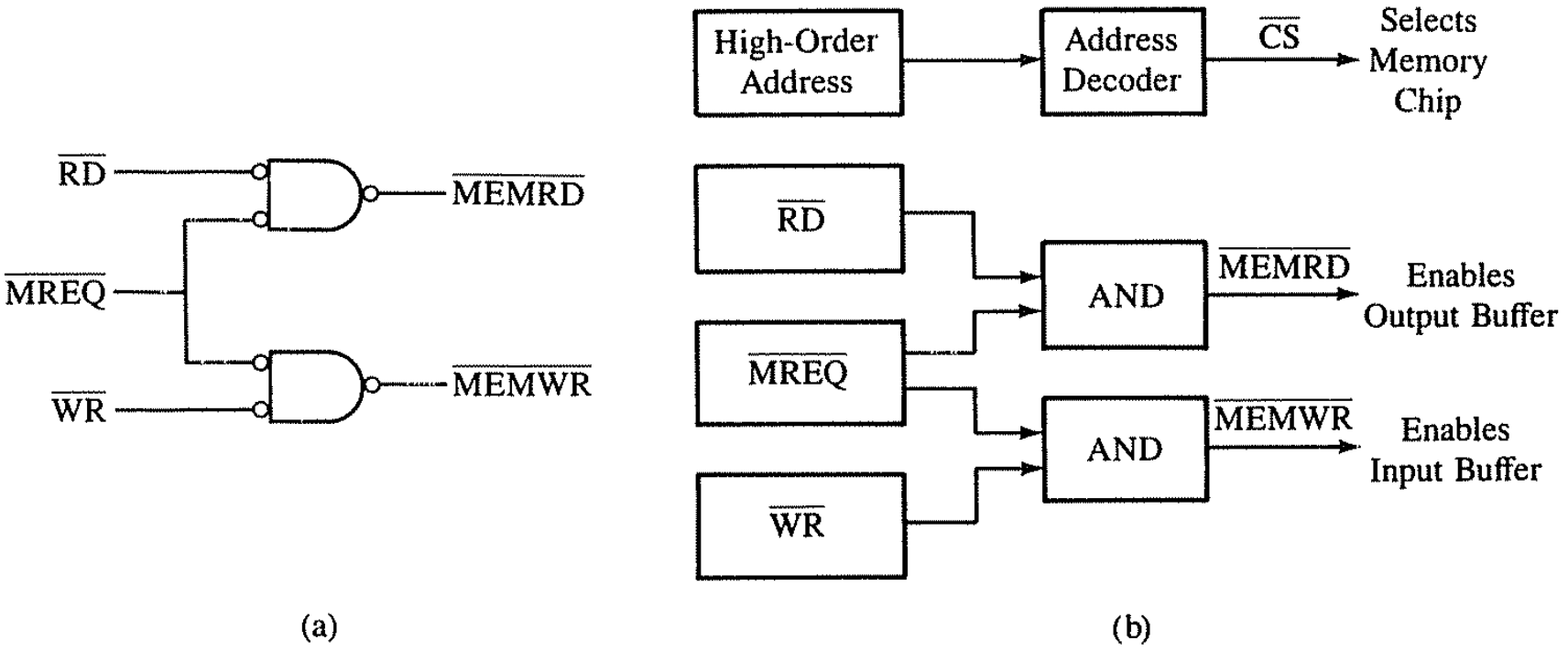


FIGURE 4.4
(a) Generating Control Signals (b) Block Diagram: Alternative Approach to Memory Read/Write Operations

برای ارتباط دهی حافظه با ریزپردازنده می توان قدم های فوق را به صورت زیر خلاصه نمود:

۱- خطوط آدرس مورد نیاز از گذرگاه آدرس را به خطوط آدرس تراشه حافظه متصل کنید.

۲- بقیه خطوط آدرس از گذرگاه آدرس را برای تولید سیگنال انتخاب تراشه ('CS) استفاده کنید.

۳- با ترکیب سیگنال 'CS و 'MREQ سیگنال انتخاب حافظه ('MESL) را تولید و از آن برای انتخاب تراشه حافظه استفاده کنید.

۴- سیگنال های کنترلی 'RD و'WR ریزپردازنده را به پایه های 'RD و 'WR تراشه حافظه جهت فعال سازی بافرها متصل کنید.

۵- یک راه دیگر، تولید سیگنال های 'MEMRD و 'MEMWR بوسیله ترکیب نمودن سیگنال های 'RD و 'WR با 'MREQ برای فعال نمودن بافرهای مورد نظر می باشد و سیگنال 'CS نیز برای انتخاب تراشه استفاده می شود.

4.14 Address Decoding

The process of **address decoding** should result in identifying a register with a given address; we should be able to generate a unique pulse for that address. For example, in Figure 4.5(a), the output of the NAND gate goes low (active) only when the address on the address lines is $F7_H$; no other address can cause the output of the gate to go low. This process is called decoding the address. We can also use a decoder for address decoding, as discussed below, or a PROM (Programmable Read-Only-Memory), as discussed in Chapter 16.

Figure 4.5(b) shows a 3-to-8 decoder and a 4-input NAND gate. The decoder has three enable lines—one active high and two active low. The enable line $\overline{E}_1$ is connected to address line A_3, and $\overline{E}_2$ is connected to address lines A_4–A_7 through the NAND gate. Address lines A_2, A_1, and A_0 are inputs to the decoder, and the enable line E_3 is tied high and is not being used here for decoding.

In this decoder circuit, three input lines can have eight different logic combinations

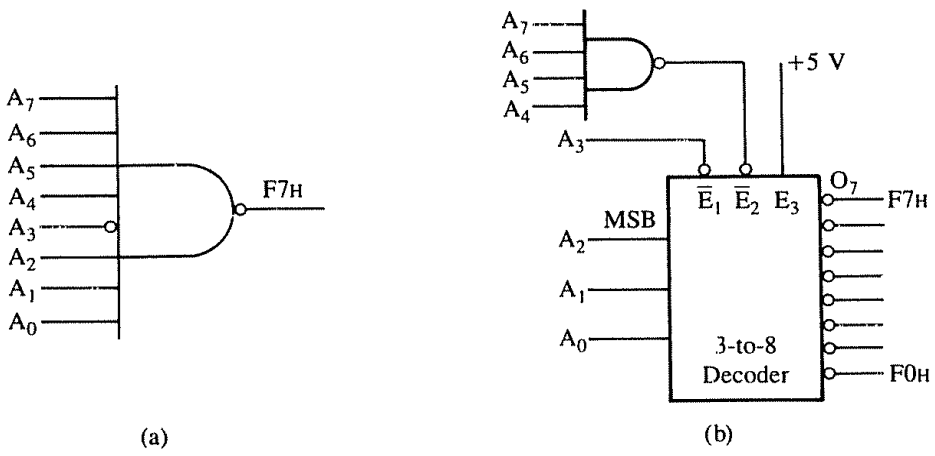
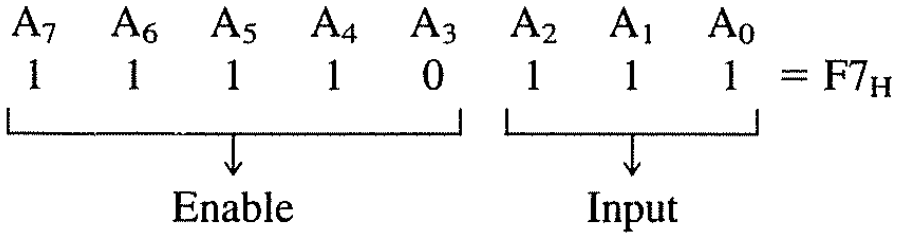


FIGURE 4.5
Address Decoding

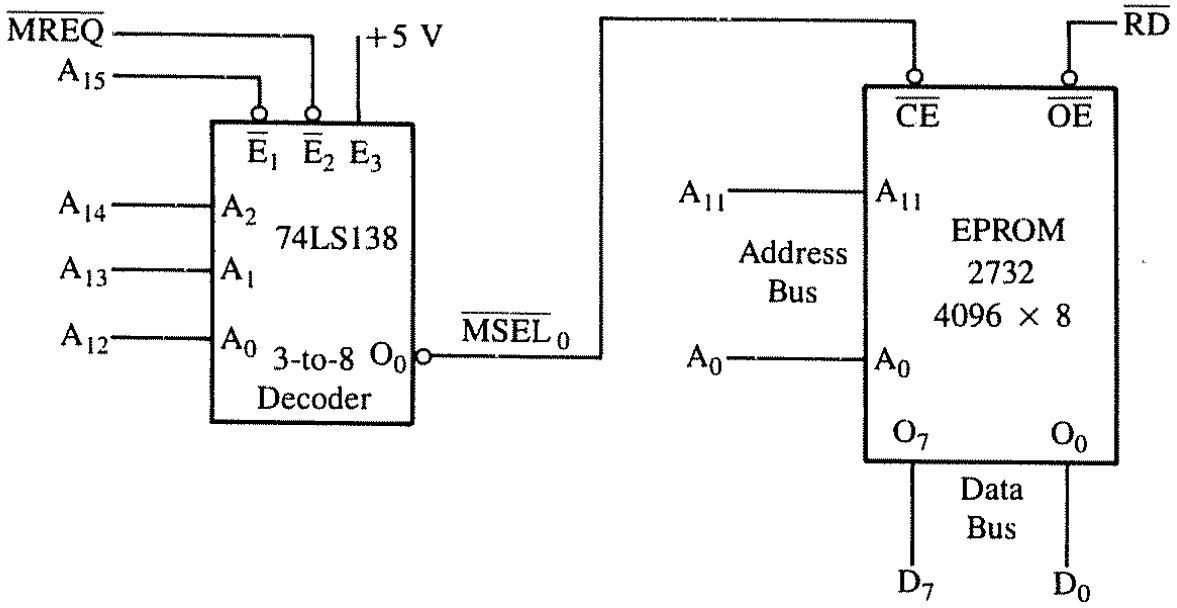
from 000 to 111; each input combination can be identified by the corresponding output line if enable lines are active. For example, if the input is 0 0 0, O_0 goes low (others remain high), and if the input is 1 1 0, O_6 goes low. To activate the enable line $\overline{E}_1$, A_3 should be low, and to activate $\overline{E}_2$, address lines A_7–A_4 should be high causing the output of the NAND gate to go low. If the input to the decoder is 1 1 1, the output line O_7 of the decoder will go low, thus decoding the address $F7_H$.



This 3-to-8 decoder can identify or decode eight addresses from $F0_H$ to $F7_H$ as

4.23 Memory Map

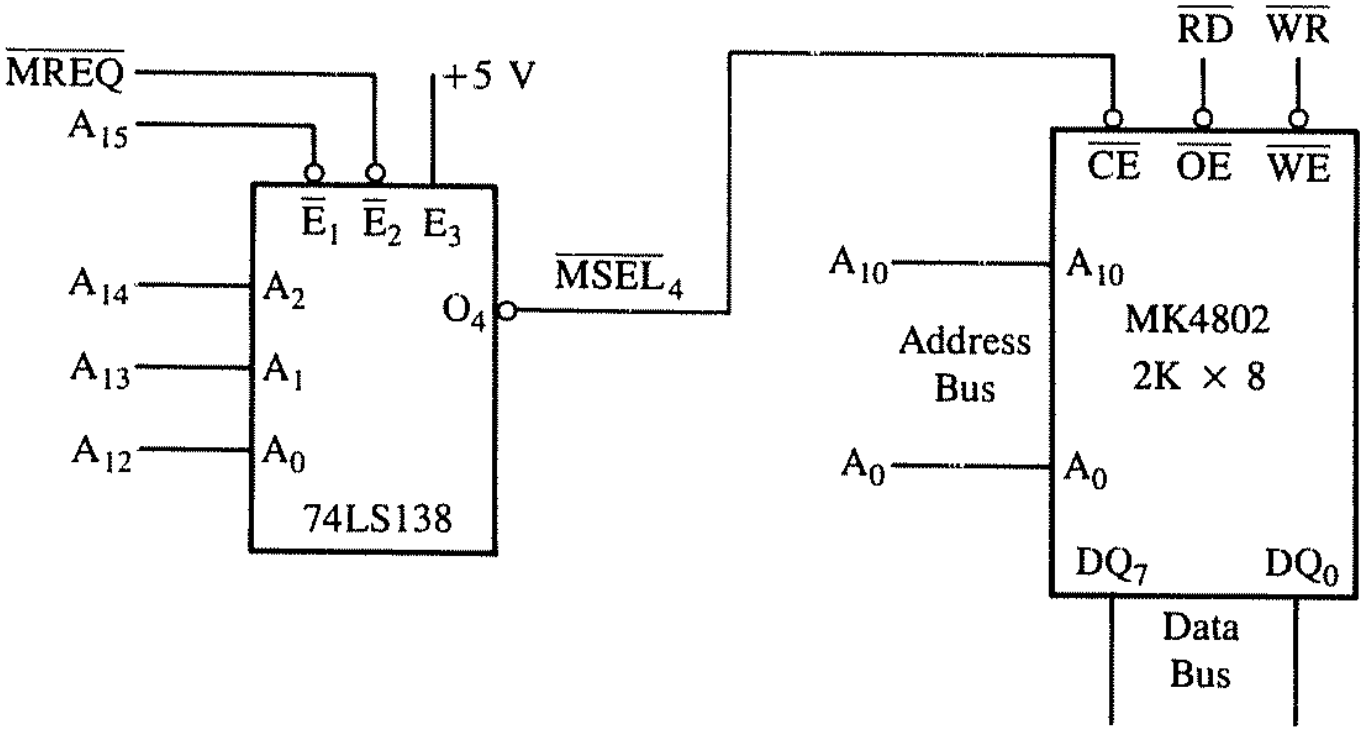
We can obtain the address range of this memory chip by analyzing the possible logic levels on the 16 address lines. The logic levels on the address lines A_{15}–A_{12} have to be 0 to assert the Chip Enable, and the address lines A_{11}–A_0 can assume any combinations from all 0s to all 1s. Therefore, the memory map of this chip ranges from 0000_H to $0FFF_H$.



A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$= 0000_H$
				↓											↓	
$\overline{MSEL_0}$				1	1	1	1	1	1	1	1	1	1	1	1	$= 0FFF_H$

Fold back Memory

(A11 not used)



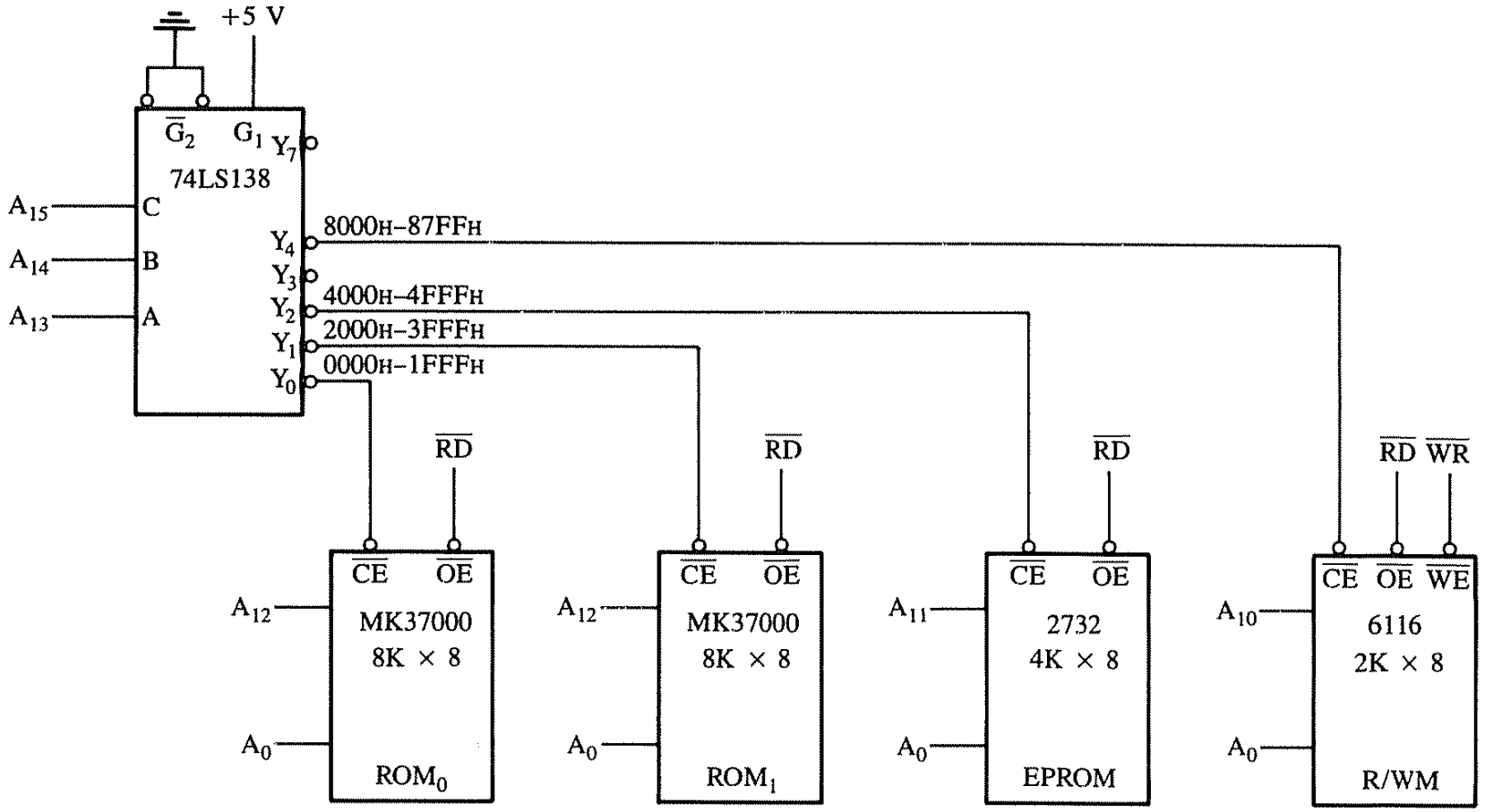
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	1	0	0	X	0	0	0	0	0	0	0	0	0	0	0	$= 4000_H$
					↓										↓	
	$\overline{MSEL_4}$			X	1	1	1	1	1	1	1	1	1	1	1	$= 47FF_H$

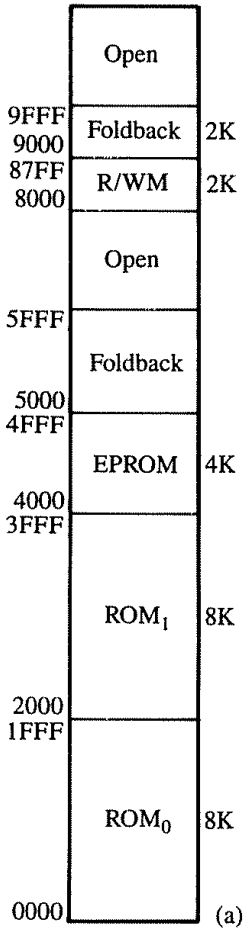
If we assume A_{11} at logic 1, the memory map ranges from 4800_H to $4FFF_H$ as shown below:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	$= 4800_H$
				↓											↓	
	$\overline{MSEL_4}$			1	1	1	1	1	1	1	1	1	1	1	1	$= 4FFF_H$

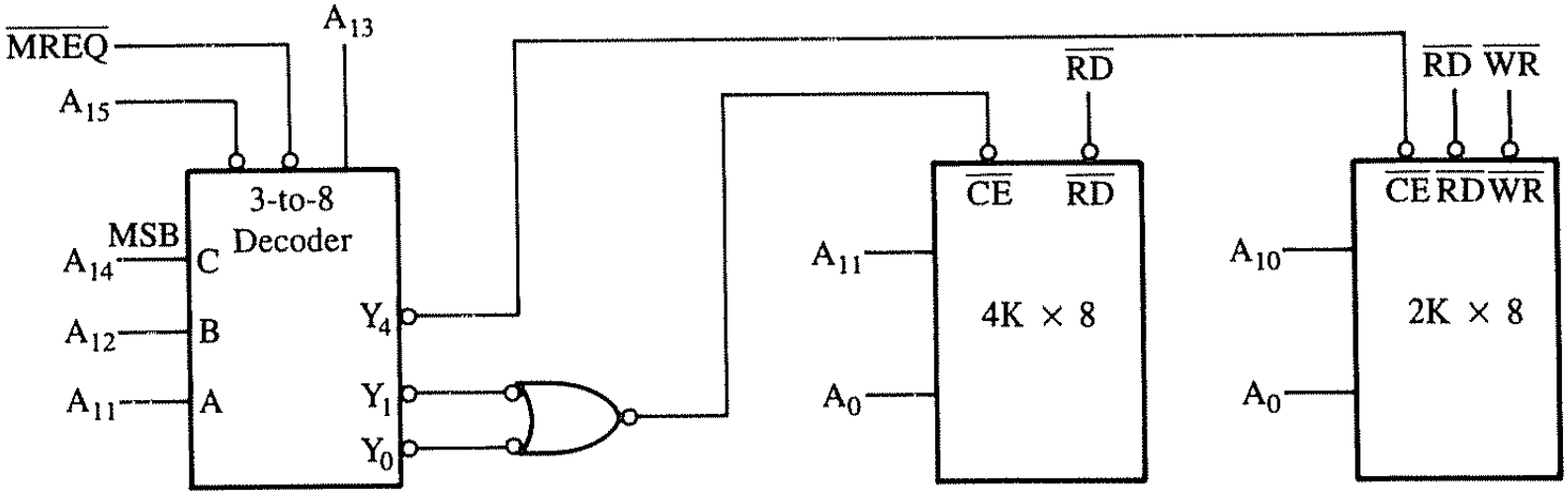
The entire memory map appears to be from 4000_H to $4FFF_H$ (4K memory). In reality, we have only 2K memory occupying the memory space of 4K. Because of the one "don't care" line, each register can have two addresses. For example, the addresses 4000_H and 4800_H will select the same register. The duplicate or redundant range of the memory addresses (4800_H to $4FFF_H$) is generally known as the **foldback memory;** this memory space cannot be used by any other memory chip.

مثال: **نقشه حافظه تراشه های حافظه مدار زیر را بدست آورید.**




Open
9FFF
9000
Foldback
2K
87FF
8000
R/WM
2K
Open
5FFF
Foldback
5000
4FFF
EPROM
4K
4000
3FFF
ROM1
8K
2000
1FFF
ROM0
8K
0000
(a)

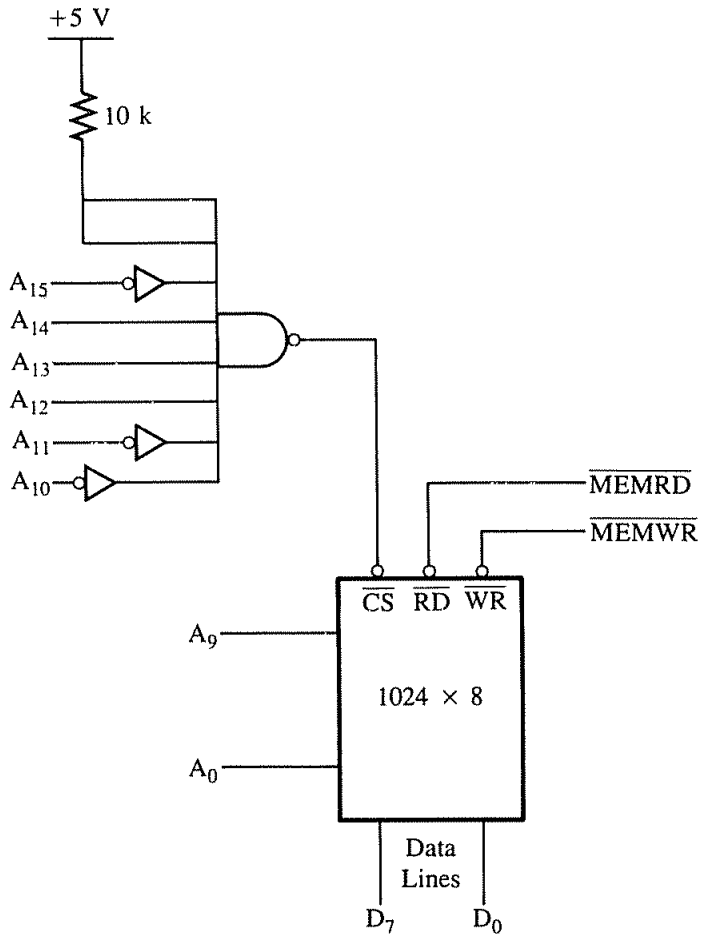
تمرین۱ : نقشه حافظه تراشه های حافظه مدار زیر را بدست آورید.



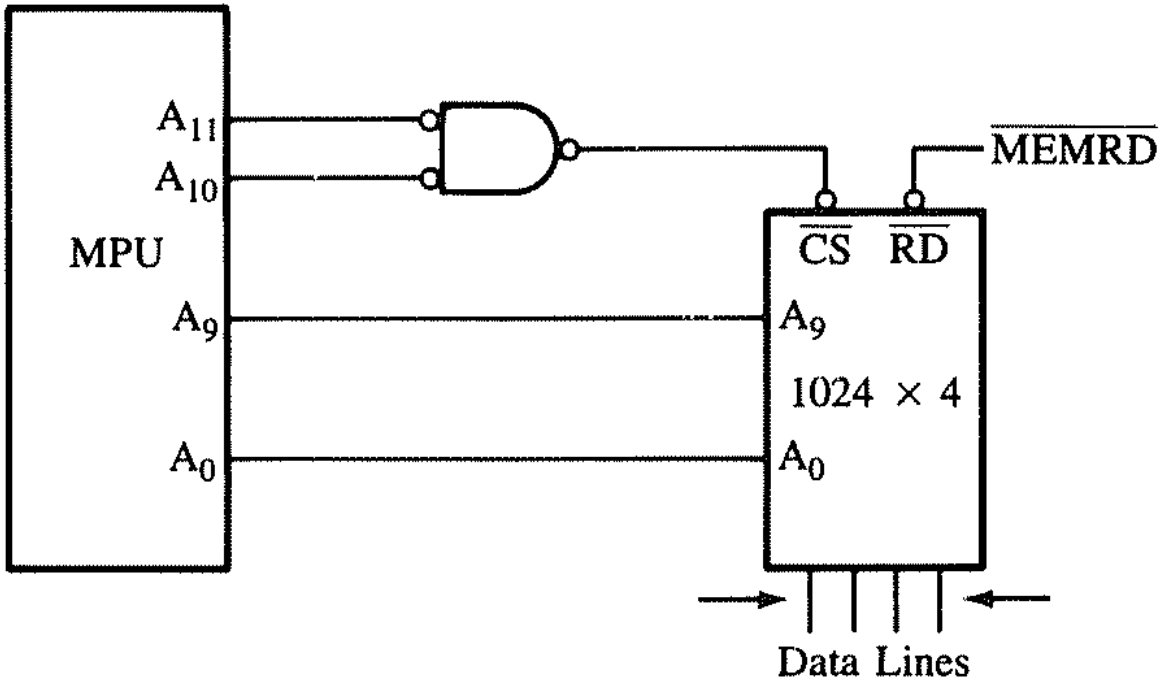
تمرین۲ : الف) محدوده آدرس تراشه حافظه زیر را بدست آورید.

ب) این تراشه چند صفحه (page) را شامل می شود.

ج) اگر گیت not از خط آدرس ۱۵ حذف شود، محدوده آدرس تراشه حافظه چه خواهد شد.



تمرین۳ – نقشه حافظه مدار زیر را ترسیم کنید.



طبقه بندی حافظه ها

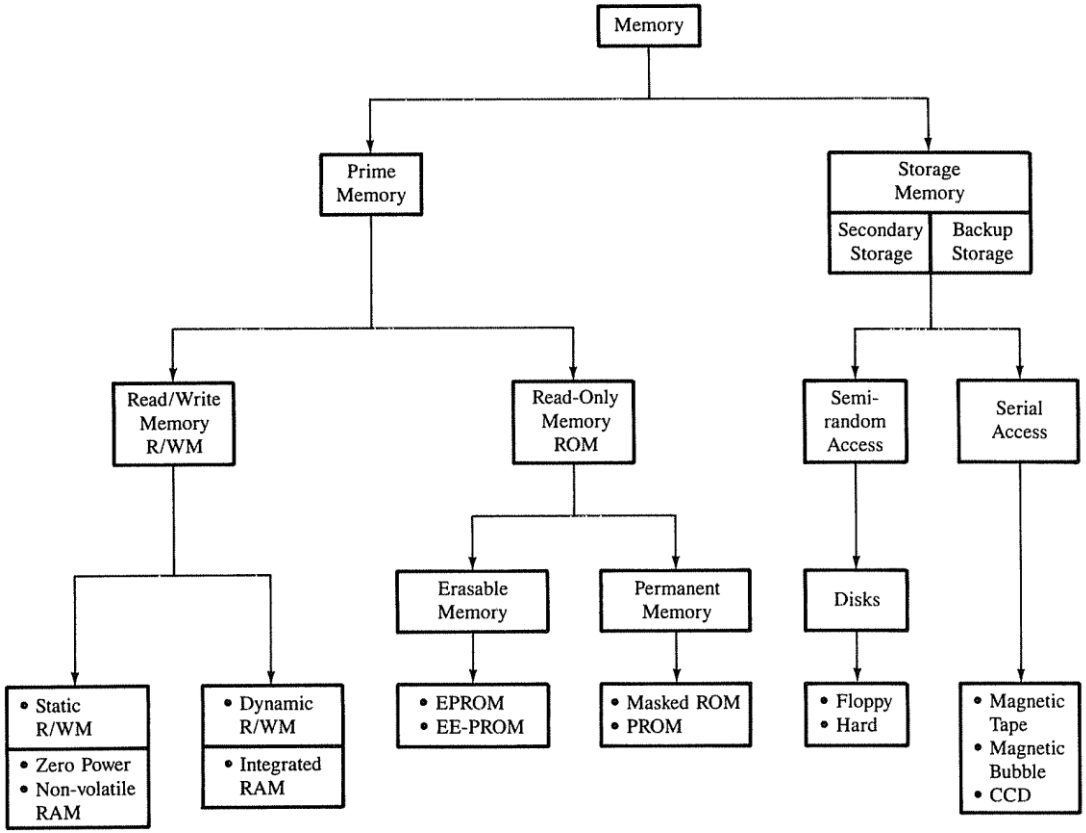


FIGURE 2.12
Memory Classification

وسایل ورودی / خروجی (I/O)

- وسایل ورودی/خروجی یا I/o رابط MPU با دنیای خارج می باشد.
- MPU داده های باینری را از طریق وسایل ورودی مانند صفحه کلید دریافت می نماید و همچنین داده ها را به وسایل خروجی مانند دیود های نوری یا چاپگر ارسال می کند.
- از دو طریق متفاوت MPU می تواند وسایل ورودی/خروجی را تشخیص دهد.
- یک روش استفاده از آدرس ۸ بیتی است و روش دیگر استفاده از آدرس ۱۶ بیتی می باشد.

I/O با آدرس ۸ بیتی (Peripheral-Mapped I/O)

- در این روش، ریزپردازنده از ۸ خط آدرس برای تشخیص وسایل ورودی و خروجی استفاده می نماید. این روش را Peripheral-mapped I/O نیز می نامند. ۸ خط آدرس ۲۵۶ (2^8) آدرس را شامل می شود.
- ریزپردازنده می تواند ۲۵۶ وسیله ورودی و ۲۵۶ وسیله خروجی را در محدوده آدرس 00h تا FFh تشخیص دهد.
- این وسایل از طریق خطوط کنترلی خواندنI/o (I/o Read) برای وسایل ورودی و نوشتنI/o (I/o Write) برای وسایل خروجی از یکدیگر جدا می شوند.

- کل محدوده آدرس I/o از 00h تا FFh را نقشه I/O (I/O Map) می نامند و هر آدرس تکی در این محدوده، آدرس وسیله I/o یا شماره پورت(Port) نامیده می شود.

در وسایل ورودی/خروجی ما باید از وسایل ارتباط دهی خارجی (external interfacing device) استفاده کنیم.

برای ارتباط با وسایل ورودی/خروجی مراحلی مشابه با ارتباط با حافظه باید در نظر گرفت که عبارتند از :

۱- ریزپردازنده آدرس ۸ بیتی را بر روی گذرگاه آدرس قرار داده که توسط مدار رمزگشای خارجی (بعدا توضیح داده می شود) رمز گشایی می گردد.
۲- ریزپردازنده سیگنال های کنترلی (I/O read or I/O write) را برای فعال نمودن وسایل ورودی/خروجی ارسال می نماید.
۳- داده ها به گذرگاه داده منتقل می شوند.

I/O با آدرس ۱۶ بیتی (Memory – Mapped I/O):

- در این روش MPUاز ۱۶ خط آدرس برای پیدا نمودن وسایل ورودی/خروجی استفاده میکند و هر وسیله ورودی/خروجی مانند یک ثبات محل حافظه در نظر گرفته می شود.

- MPUاز سیگنال های مشابه Memory readیا Memory write و دستورات مشابه ایی که برای ثبات های حافظه استفاده می گردید استفاده میکند. زیرا که دسترسی به هر وسیله ورودی/خروجی را مانند دسترسی به یک ثبات حافظه ایی تلقی می نماید

مثالی از یک سیتم میکروکامپیوتر

مطالب مطرح شده را می توان در شکل 2.13 خلاصه نمود:

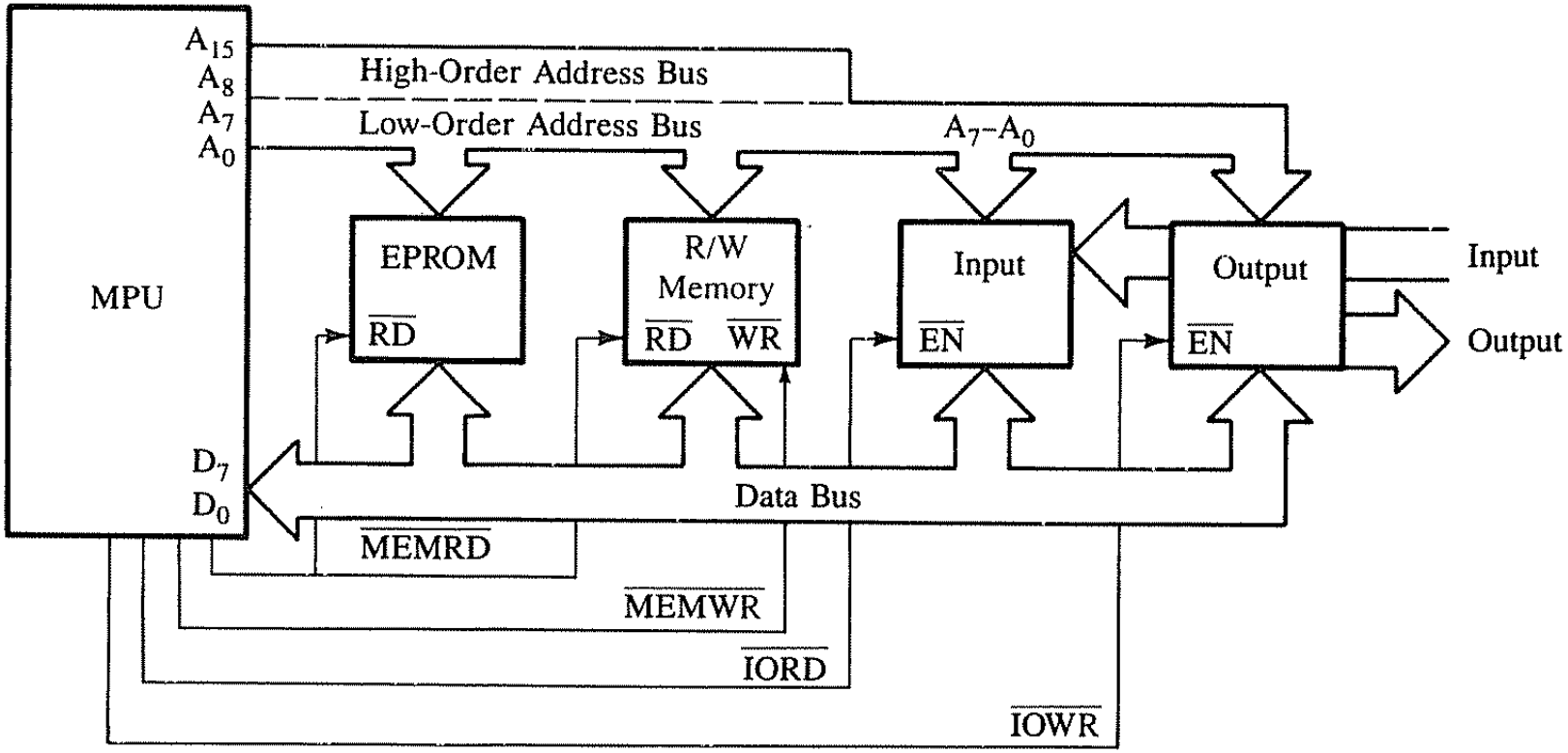


FIGURE 2.13
Example of a Microcomputer System

برای درک عملکرد سیستم نشان داده شده برنامه ایی با سه دستورالعمل که از قبل در داخل حافظه RAMذخیره گردیده است را در نظر بگیریم:

دستورات عبارتند از

۱- پورت ورودی شماره 20h خوانده شود.

۲- داده را از طریق پورت خروجی شماره 80h نمایش دهید.

۳- توقف برنامه (Stop)

برای اجرای این دستورات، MPUعملیات زیر را انجام می دهد:

۱- آدرس محل حافظه ایی که دستور ۱ در آن قرا دارد مشخص و با استفاده از سیگنال کنترلی خواندن حافظه‘(MEMRD)دستور فراخوانی می گردد. دستور رمز گشایی می گردد:

با قرار گرفتن آدرس پورت ورودی و استفاده از سیگنال کنترلی خواندن '(IORD) محتویات پورت ورودی خوانده و در داخل یک از ثبات ها ذخیره می گردد.

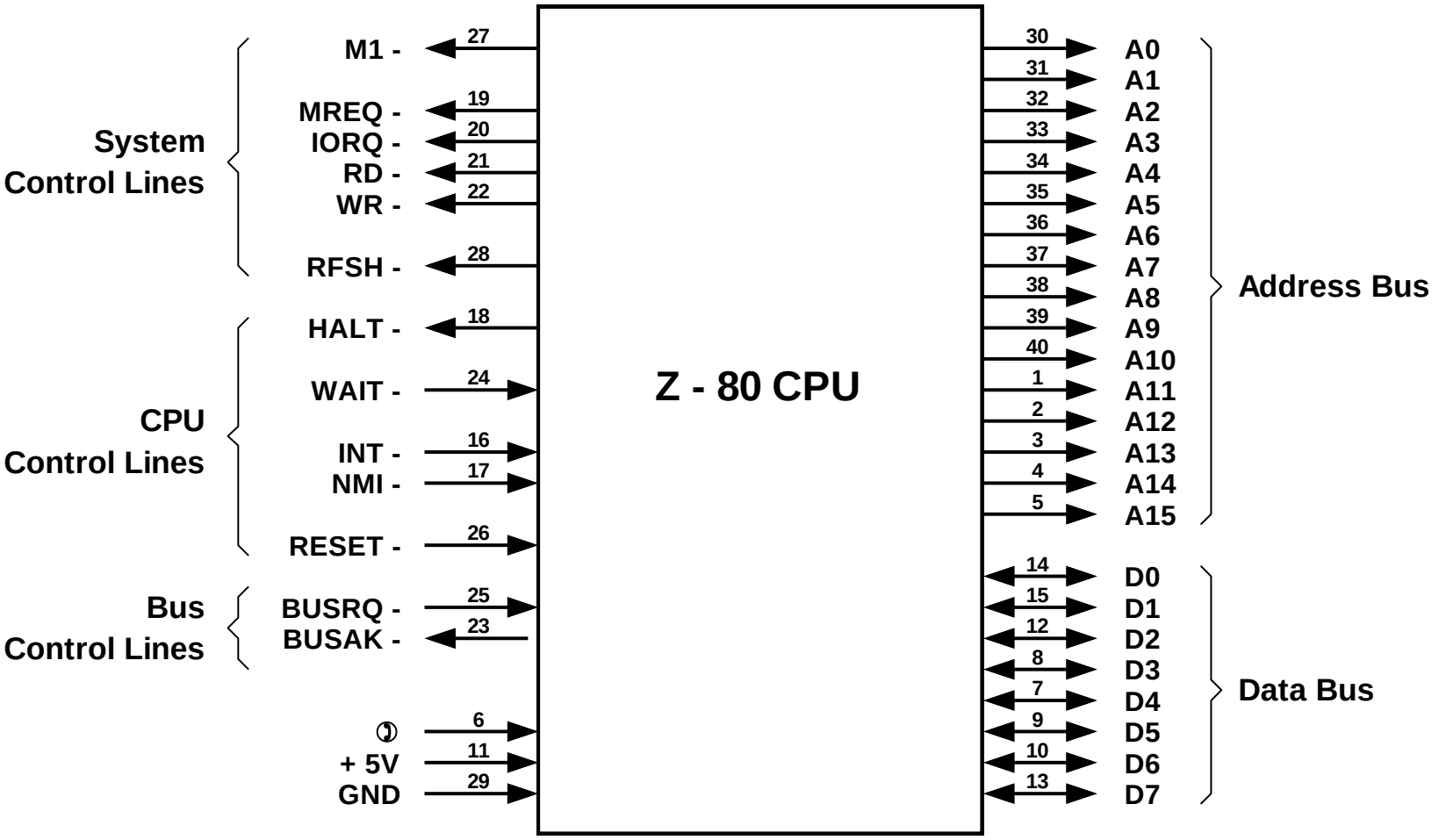
۲- با قرار دادن آدرس و سیگنال کنترلی 'MEMRD دومین دستور فراخوانی می گردد و سپس دستور رمز گشایی می شود:

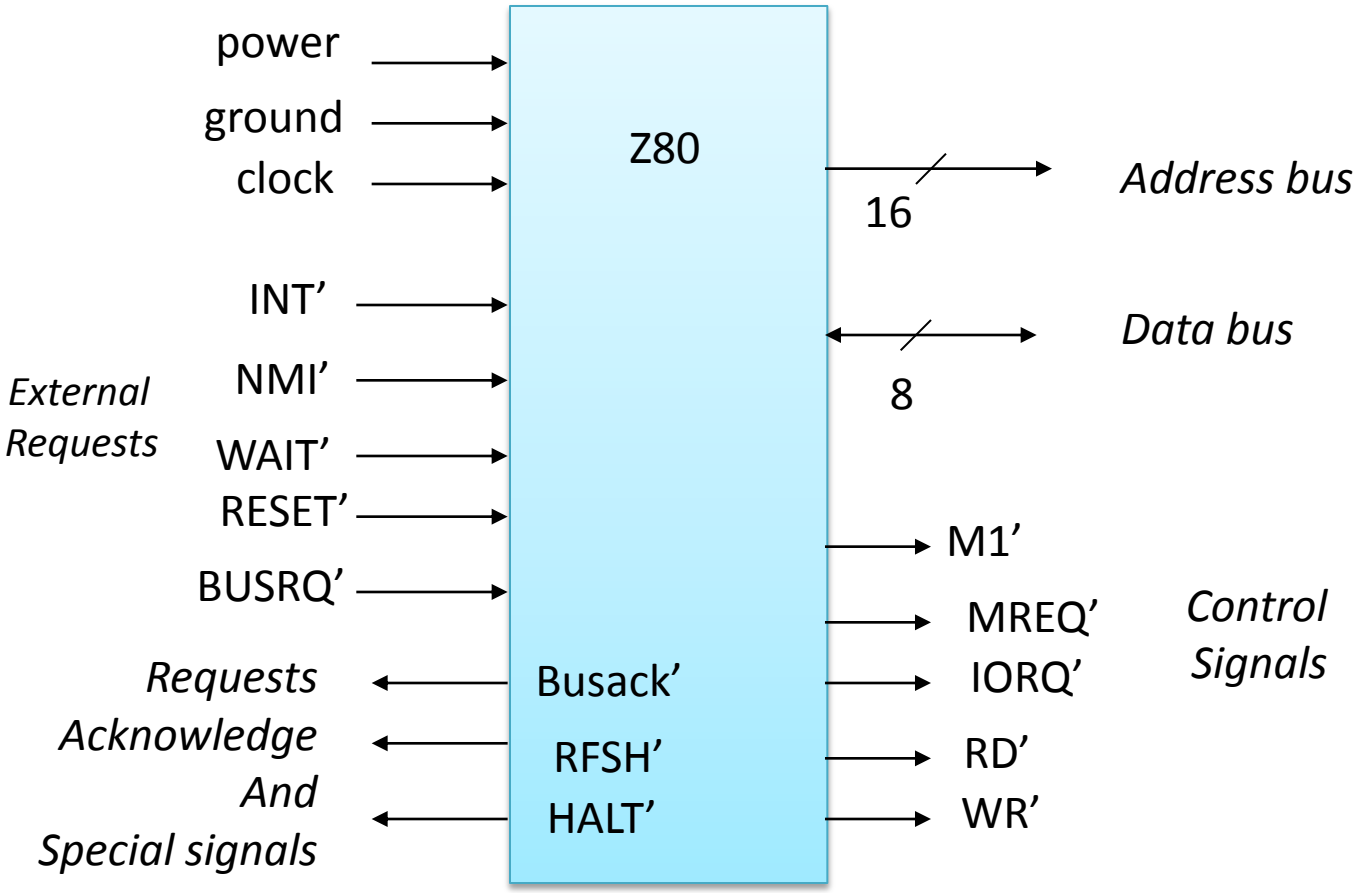
با قرار گرفتن آدرس پورت 80h و سیگنال کنترلی 'IOWR داده منتقل می شود.

۳- با فراخوانی آخرین دستور از حافظه و رمز گشایی آن، برنامه متوقف می شود.

مطالب مطرح شده شکل خلاصه شده ایی است از اینکه سیستم چگونه کار میکندا، البته این توضیحات شامل جزئیاتی مانند دستورات چند بایتی و سیکل های ماشینی و زمانبندی نمی باشد

Z80 – pin out


Z - 80 CPU
System Control Lines
CPU Control Lines
Bus Control Lines
M1 - 27
MREQ - 19
IORQ - 20
RD - 21
WR - 22
RFSH - 28
HALT - 18
WAIT - 24
INT - 16
NMI - 17
RESET - 26
BUSRQ - 25
BUSAK - 23
6
+ 5V 11
GND 29
30 A0
31 A1
32 A2
33 A3
34 A4
35 A5
36 A6
37 A7
38 A8
39 A9
40 A10
1 A11
2 A12
3 A13
4 A14
5 A15
Address Bus
14 D0
15 D1
12 D2
8 D3
7 D4
9 D5
10 D6
13 D7
Data Bus


power
ground
clock
Z80
16
Address bus
INT'
Data bus
External
Requests
NMI'
8
WAIT'
RESET'
M1'
BUSRQ'
MREQ'
Control
Signals
Requests
Busack'
IORQ'
Acknowledge
RFSH'
RD'
And
HALT'
WR'
Special signals

مدل برنامه نویسی Z80

- در این مدل قسمتهایی که برای یک برنامه نویس لازم می باشد لحاظ گردیده است.

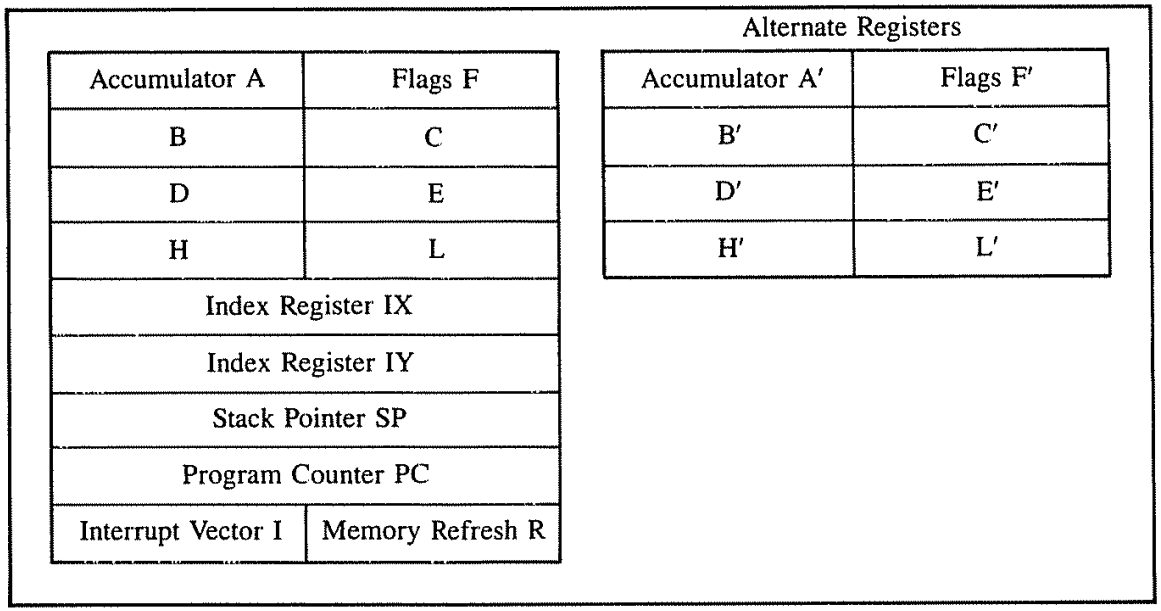


FIGURE 3.1
The Z80 Programming Model

انباره (Accumulator)

- انباره یک ثبات ۸ بیتی است که قسمتی از واحد ریاضی و منطق (ALU) بوده و با ثبات A مشخص می شود.

- از این ثبات برای ذخیره نمودن داده ۸ بیتی و عملیات ریاضی و منطقی استفاده می شود. نتیجه دستوری که در ALU انجام شده است در این ثبات ذخیره می گردد.

- برای مثال، در یک جمع ۸ بیتی ، یکی از دو داده همیشه در ثبات انباره و نتیجه عمل جمع هم در نهایت در انباره ذخیره می گردد و جایگزین داده قبلی می شود.

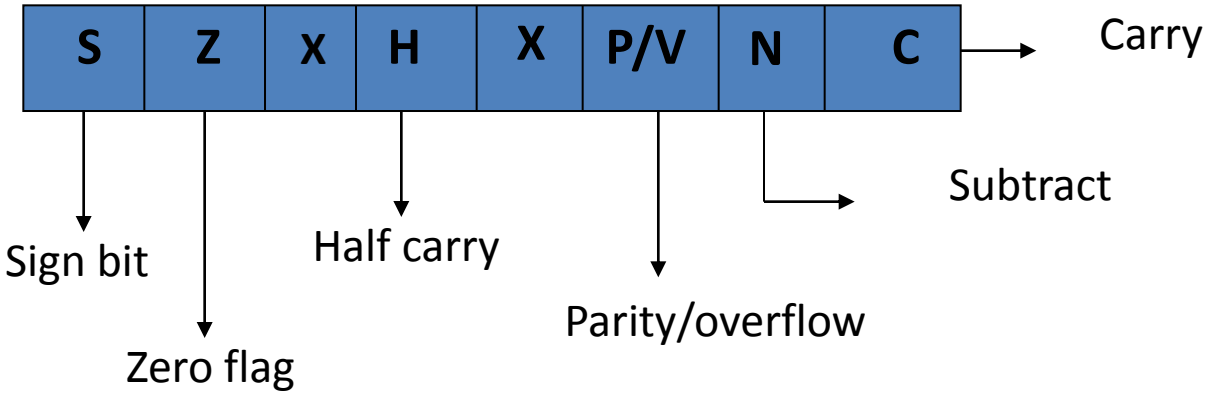
ثبات پرچم (Flag Register)

- ALUدارای ۶ فلیپ فلاپ می باشد که طبق نتیجه بدست آمده بعد از هر عمل ALU یک (Set)یا صفر (Reset)می گردند.

- وضعیت هر فلیپ فلاپ به نام پرچم در ثبات پرچم (F) نشان داده می شود. وضـعیت هر یک از ۶ پرچم در ثباتی ۸ بیتی بنام ثبات پرچم ذخیره گردیده و در صورت نیـاز مورد بررسی قرار خواهد گرفت.

ثبات پرچم Flag Register

این ثبات برای حفظ و نگهداری اطلاعات مربوط به نتایج حاصل از عملیات مختلف ریزپردازنده می باشد.

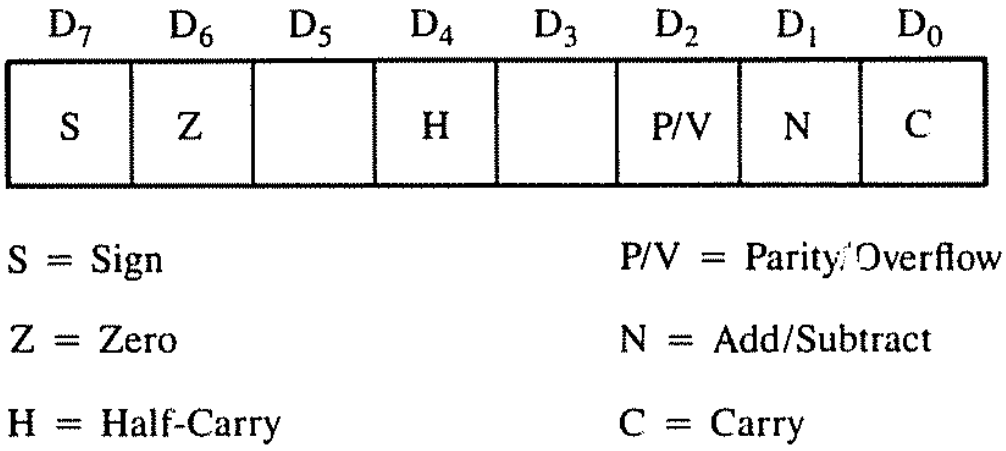


کاربرد عمده بیتهای پرچم در دستورات پرش شرطی می باشد.

کاربرد بیتهای H و N در محاسبا ت BCD میباشد.

محل قرار گرفتن هر پرچم در ثبات پرچم در شکل زیر نشان داده شده است. بیت های ۳ و ۵ استفاده نشده اند.

FIGURE 3.2
Flag Register: Bit Identification



- در میان ۶ پرچم، پرچم های H (Half Carry) و N (Add/Subtract) بصورت داخلی و برای عملیات BCD توسط ریزپردازنده مورد استفاده قرار می گیرد و در دسترس برنامه نویس برای تصمیم گیری نمی باشند. چهار پرچم باقی مانده می توانند توسط دستورات پرش شرطی(Conditional Jump) و دستورات فراخوان (Call)برای تصمیم گیری مورد بررسی قرار بگیرند.

پرچم C (Carry Flag)

- اگر یک عمل ریاضی یک Carry در عمل جمع یا یک قرض در عمل تفریق ایجاد کند این پرچم یک می شود وگرنه صفر می شود. این مهم است که بخاطر داشته باشید وقتی یک عمل ریاضی carry تولید نکند پرچم carry صفر می شود. این پرچم توسط دستورات دیگر مانند دستور ات منطقی و دستورات Shift نیز تحت تاثیر قرار می گیرد.

- البته دو دستور SCF (Set carry flag) و
  دستور CCF (Complement carry flag)

مستقل از عمل قبلی که در ALU صورت گرفته است SET یا مکمل می گردد.

پرچم Z (Zero Flag)

- اگر نتیجه یک عمل ۸ بیتی صفر شود این پرچم یک (Set) می گـردد. وگرنـه صـفر (reset) می شود.
- در عمل چک کردن یک بیت در یک ثبات، اگر بیت مورد بررسی صفر باشد پـرچم Z یک می شود. و گرنه صفر می گردد.
- در دستور مقایسه پرچم Z یک می شـود اگـر دو عـدد مسـاوی باشـند وگرنـه صـفر می گردد.
- پرچم Z توسط چند دستور دیگر که در جدول دستورات مشخص شده اند تحت تاثیر قرار می گیرد.

پرچم S (Sign Flag)

- بعد از یک عمل ALU، اگر با ارزش ترین بیت یک شود پرچم S نیز یک می شود وگرنه پرچم S صفر می گردد.
- وقتی پرچم علامت یک میشود لزوما به معنی منفی بودن نتیجه نیست. بلکه این موضوع بستگی به سیستم عددی (اعداد بدون علامت، اعداد با علامت یا مکمل دو) که برنامه نویس استفاده می کند دارد.

پرچم P/V (Parity/Overflow)

- این پرچم به صورت دو منظوره مورد استفاده قرار می گیرد. برای چک کردن Parity (تعداد یک های موجود در یک بایت) و چک نمودن سرریز در اعداد با علامت.
- در مورد چک کردن parity، اگر تعداد یک های موجود در نتیجه حاصل از یک عمل منطقی زوج باشد این پرچم set می شود وگرنه صفر می گردد.

- برای مثال، اگر نتیجه AND نمودن دوعدد ۸ بیت باینری 00000011 شود پرچم parity یک می شود.

- در عملیات ریاضی اعداد با علامت (مکمل دو) اگر نتیجه حاصل از یک عمل ریاضی از ۱۲۷- بزرگتر و از ۱۲۸- کوچکتر باشد سرریز رخ داده است و این پرچم یک می شود وگرنه صفر می گردد.

پرچم H (Half Carry)

- در عملیات ریاضی، این پرچم بوسیله carry یا borrow بین بیت های سوم وچهارم صفر یا یک میشود.

- درعمل جمع، وقتی از بیت سوم به بیت چهارم carryوجود داشته باشد این پرچم یک می گردد، وگرنه صفر می شود.

- در عمل تفریق، وقتی از بیت چهارم به بیت سوم قرض داده می شود این پرچم یک می گردد وگرنه صفر می شود.

پرچم N (Add/Subtract Flag)

- این پرچم نیز بصورت داخلی برای عملیات BCD مورد استفاده قرار می گیرد تا بتواند بین عمل جمع و تفریق تفاوت قائل شود.
- برای عملیات BCD این پرچم صفر است و برای عمل تفریق این پرچم یک می شود.

ثبات های همه منظوره و ثبات های رزرو

- ریزپردازنده Z80 دارای ۶ ثبات ۸ بیتی همه منظوره به نام های B,C,D,E,H,L می باشد.
- این ثبات های ۸ بیتی برای ذخیره نمودن داده ها در طول اجرای برنامه استفاده می شوند. این ثبات ها را می توان بصورت جفت ثبات های BC,DE,HL در عملیات ۱۶ بیتی یا برای نگهداری آدرس محل حافظه از آنها استفاده نمود.
- برنامه نویس می تواند از این ثباتها برای بارگذاری وکپی کردن داده ها نیز استفاده نماید.
- برای مثال دستور LD B,C محتوی ثبات C را در داخل ثبات B کپی می نماید.
- بنابراین ، از این ثباتها می توان به عنوان محل حافظه استفاده نمود و تنها تفاوت آنها در این است که در داخل ریزپردازنده قرار دارند و دارای اسامی از قبل تعیین شده و ثابتی می باشند.

- بعضی از ریزپردازنده ها این ثبات ها را نداشته و از محل های حافظه به عنوان ثبات استفاده می کنند.

- در کنار ثبات های همه منظوره ، ریزپردازنده Z80 از ۶ ثبات مشابه رزرو به نام های A',B',C',D',E',H' نیز استفاده میکند. این ثبات های ۸ بیتی صرفا برای جابجایی (Exchange)داده ها با ثبات های همه منظور بکار گرفته می شوند.

- این ثباتها بصورت مستقیم در دسترس برنامه نویس قرار ندارد، فقط از طریق دستورات جابجایی میتوان آنها را بکار گرفت.

ثبات های ۱۶ بیتی به عنوان اشاره گر حافظه

- ریزپزدازنده Z80 دارای چهار ثبات ۱۶ بیتی برای نگهداری آدرس محل حافظـه هـا می باشد. این ثبات ها به عنوان اشاره گرهای حافظه طبقه بندی گردیده اند.

- عمل اصلی حافظه، نگهداری داده ها و دستورالعمل ها می باشد و ریزپردازنده ها نیاز دارند که به این داده ها و دستورالعمل ها دسترسی پیدا کرده و آنها را بخوانند. بـرای دسترسی به ثبات های حافظه(محل هـای حافظـه)، ریزپردازنـده محـل حافظـه را از طریق آدرس های موجود در این ثبات های اشاره گر حافظه ۱۶ بیتی پیدا می کند.

ثبات شاخص (Index Register)

- Z80 دارای دو ثبات شاخص ۱۶ بیتی به نام های IX وIY می باشد. هر یک از این ثبات ها برای نگهداری آدرس حافظه (که از حاصل جمع آدرس ۱۶ بیتی و عدد جابجایی (OFFSET) ۸ بیتی بدست می آید) بکار می روند.

- برای مثال اگر ثبات IX حاوی عدد 2050h باشد و نیاز به دسترسی به محل 2060h باشد باید عدد جابجایی 10h را با عدد 2050h جمع نمود.

- بصورت مشابه اگر ادرس 2040h مورد نظر باشد باید عدد 10H- را بصورت مکمل دو با عدد موجود در ثبات شاخص جمع نمود.

- از جفت ثبات HL نیز بعضی وقتها به عنوان اشاره گر حافظه نیز می توان استفاده نمود. البته بدون استفاده از عدد اضافی جابجایی.

Stack Pointer (SP)

- SP همچنین یک ثبات ۱۶ بیتی برای اشاره به محل های حافظه در Stack می باشد.
- Stack بخشی از حافظه RAM بوده و همیشه آدرس شروع Stack در داخل SP قرار می گیرد. مفهوم کامل Stack بعدا توضح داده خواهد شد.

Program Counter (PC)

- این ثبات به صورت یک شمارنده ۱۶ بیتی کار میکند. ریزپردازنده از این ثبات به عنوان ترتیب اجرای دستورات استفاده می کند.
- PC به آدرس محل حافظه ایی که دستور بعدی در آن قرار دارد و باید فراخوانی شود اشاره دارد. وقتی ریزپردازنده آدرس مورد نظر را جهت فراخوانی دستور برروی گذرگاه آدرس قرار داد به محتویات Pc یکی اضافه میگردد تا به آدرس بعدی اشاره نماید.

ثبات های خاص (Special –purpose registers)

Z80 دارای دو ثبات با عملکرد خاص مـی باشـد کـه معمـولا در ریزپردازنـده هـای دیگـر پیـدا نمی شوند.

ایـن ثبـات هـا بـا اسـامی Interrupt Vector(I) و memory refresh register(R) می باشند.

Interrupt Vector Register (I)

این ثبات ۸ بیتی در فرآیند وقفه بکار می رود. وقتی یک دستگاه خارجی دستور وقفـه را بـه ریزپردازنده ارسال می دارد، ریزپردازنده باید کار جاری خود را به صورت تعلیق در آورد و بـه محلی از حافظه برود که مورد درخواست وسیله خارجی می باشد.

Memory Refresh Register(R)

این ثبات ۸ بیتی به صورت یک شمارنده ۷ بیتی برای نگهداری آدرس محل هایی از حافظه RAMدینامیکی که باید مورد بازسازی قرار گیرند استفاده می شود.

اجرای دستور

سیکل های ماشینی (Machine cycles)

- ریزپردازنده Z80 طوری طراحی گردید تا بتواند ۱۵۸ دستورالعمل مختلف را اجرا نماید.
- هر دستورالعمل از دو قسمت تشکیل شده است: کد عملیاتی (Opcode) و عملوند (Operand).
- Opcode فرمانی است که نوع عمل را مشخص میکند مانند ADD
- و Oprand یک بایت داده یا محتویات ثباتی است که عملیات برروی آن انجام می گیرد.
- بعضی از دستورات ، دستورات یک بایتی و بعضی دیگر دستورات چند بایتی هستند.

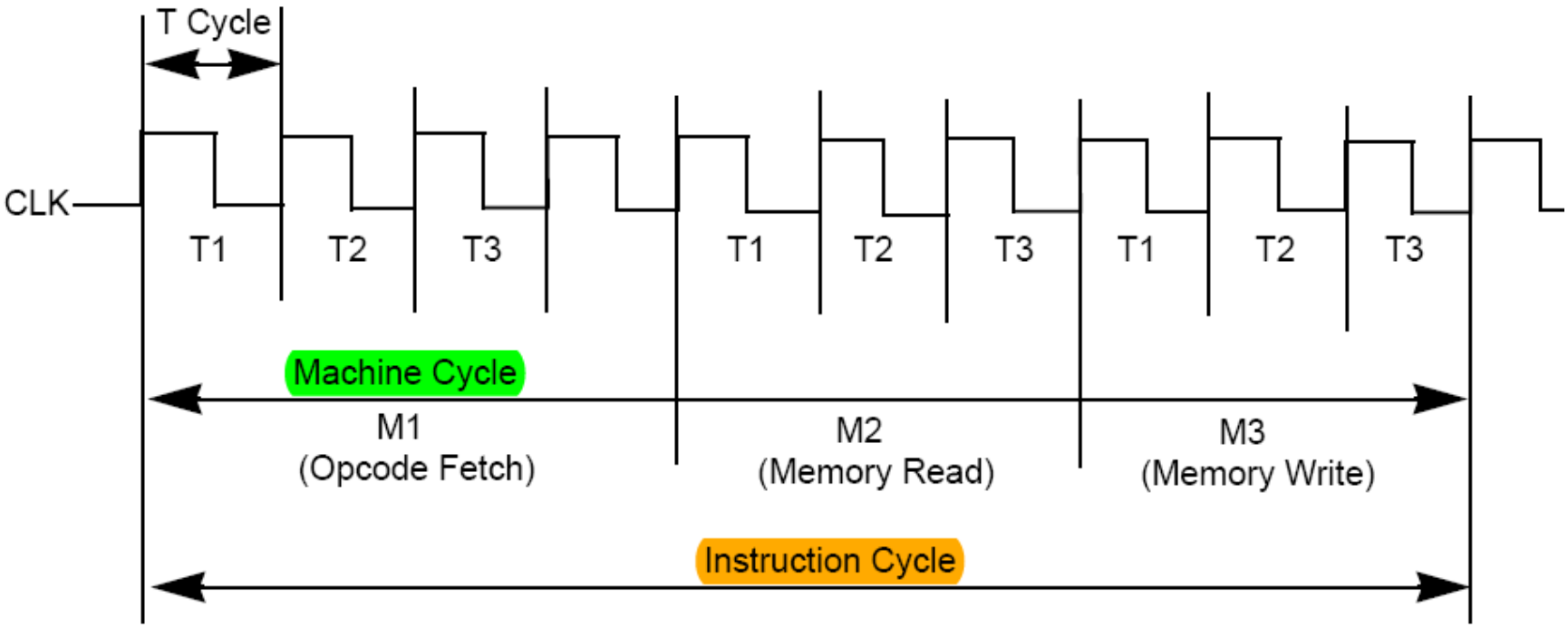
- برای اجرای یک دستور، ریزپردازنده باید عملیات مختلفی را انجام دهد، مانند Memory Read/Write و I/O Read/Write

- رابطه مستقیمی بین تعداد بایت های تشکیل دهنده ی یک دستور و تعداد عملیلاتی که برای اجرای همان دستور باید انجام گیرد وجود ندارد.

- برای مثال، دستوری که محتویات انباره را به پورت خروجی شماره ۱۰ ارسال میکند یک دستور دو بایتی است: OUT (10h), A

- بایت اول : OUT، (OPcode) دستور: به معنی خارج نمودن داده است .
- بایت دوم: (10h),A ، (Operand)دستور : بایتی که باید از طریق انباره به پورت شماره ۱۰ ارسال شود.
- اما Z80 برای اجرای این دستور باید سه عمل انجام دهد: (۱) خواندن بایت اول از حافظه، (۲) خواندن بایت دوم از حافظه و (۳) ارسال داده به پورت شماره ۱۰ .

هر دستورالعمل به چند عمل اصلی به نام سیکل ماشینی و هر سیکل ماشینی به چند پریود زمانی یا سیکل ساعت (Clock Cycle)تقسیم می شوند.



عملیات ارتباطی خارجی ریزپردازنده می تواند در سه گروه اصلی قرار گیرد:

۱- خواندن و نوشتن حافظه

۲- خواندن و نوشتن ورودی/خروجی

۳- تایید درخواست

این عملیات خود به عملیات مختلفی (سیکل ماشینی) تقسیم میشوند که در جدول۳,۱ نشان داده شده است. هر دستور به یک تا چند سیکل ماشینی و هر سیکل ماشینی نیز به چند سیکل ساعت (پریود) تقسیم شده است

TABLE 3.1
The Z80 Machine Cycles and Control Signals

Machine Cycle	$\overline{M_1}$	$\overline{MREQ}$	$\overline{IORQ}$	$\overline{RD}$	$\overline{WR}$
Opcode Fetch ($\overline{M_1}$)	0	0	1	0	1
Memory Read	1	0	1	0	1
Memory Write	1	0	1	1	0
I/O Read	1	1	0	0	1
I/O Write	1	1	0	1	0
Interrupt Acknowledge	0	1	0	1	1
Non-maskable Interrupt	0	0	1	0	1
Bus Acknowledge ($\overline{BUSAK} = 0$)	1	Z	Z	Z	Z

NOTE: Logic 0 = Active, Logic 1 = Inactive, Z = High Impedance

برای درک عملیات مختلف ، ما باید سه واژه سیکل دستورالعمل، سیکل ماشین و سیکل ساعت را تعریف کنیم:

سیکل دستورالعمل: مدت زمان مورد نیاز برای اجرای کامل یک دستورالعمل.
این سیکل می تواند خود از یک تا ۶ سیکل ماشینی تشکیل شده باشد.

سیکل ماشینی (Machine cycle) : مدت زمان لازم برای تکمیل نمودن یک عمل دسترسی به حافظه یا I/O و یا تایید یک درخواست خارجی (Acknowledging an external Request) .
این سیکل می تواند خود از ۳ تا ۶ سیکل ساعت تشکیل شده باشد.

سیکل ساعت(Clock cycle) : یک پریود کامل که از روی فرکانس سیستم محاسبه می شود.

فراخوانی Opcode سیکل ماشینی '(M1)

- اولین عمل در اجرای هر دستور فراخوانی Opcode می باشد.

مثال زیر را درنظر بگیرید:

ثبات انباره Z80 حاوی عدد 9Fh می باشد، و کد ماشینی دستور LD B, A عدد 01000111 (47H) در محل حافظه با آدرس 2002hذخیره گردیده است. این یک دستو یک بایتی می باشد و وقتی که این دستور اجرا شود محتویات انباره به داخل ثبات B کپی می شود.

تمرین: توالی اتفاقاتی که صورت می گیرد تا این کد ماشین اجرا گردد و سیگنال های مختلفی که برروی گذرگاه های سه گانه طبق ساعت سیستم رخ میدهد را نشان دهید.

حل: قبل از اینکه ریزپردازنده بتواند opcodeرا اجرا کند، باید آن را از حافظه فراخوانی نماید.
برای فراخوانی OPcodeریزپردازنده باید کارهای زیر را انجام دهد:

۱- ریزپردازنده محتویات شمارنده برنامه (2002h)را برروی گذرگاه آدرس قرار داده و سپس محتوی درون PCرا یکی اضافه میکند تا به دستور بعدی اشاره کند. PCهمیشه به دستور بعدی که قرار است اجرا شود اشاره می کند.

۲- آدرس توسط مدار رمز گشای خارجی رمزگشایی و سپس محل حافظه(ثبات) 2002hشناسایی می گردد.

۳- ریزپردازنده سیگنالهای کنترلی('MREQ','RD') را برای فعال نمودن بافر خروجی حافظه ارسال می کند.

۴- محتوی محل حافظه (opcode 47h) برروی گذرگاه داده قرار می گیرد و به درون ثبات دیکودر دستورالعمل ریزپردازنده منتقل می شود.

۵- ریزپردازنده دستور دریافتی را رمزگشایی و سپس اجرا می نماید. یعنی محتویات انباره را در داخل ثبات B کپی می نماید.

شکل 3.5 نشان می دهد که چگونه ریزپردازنده با استفاده از گذرگاه های آدرس، داده و سیگنال های کنترلی OPcode را فراخوانی می کند.

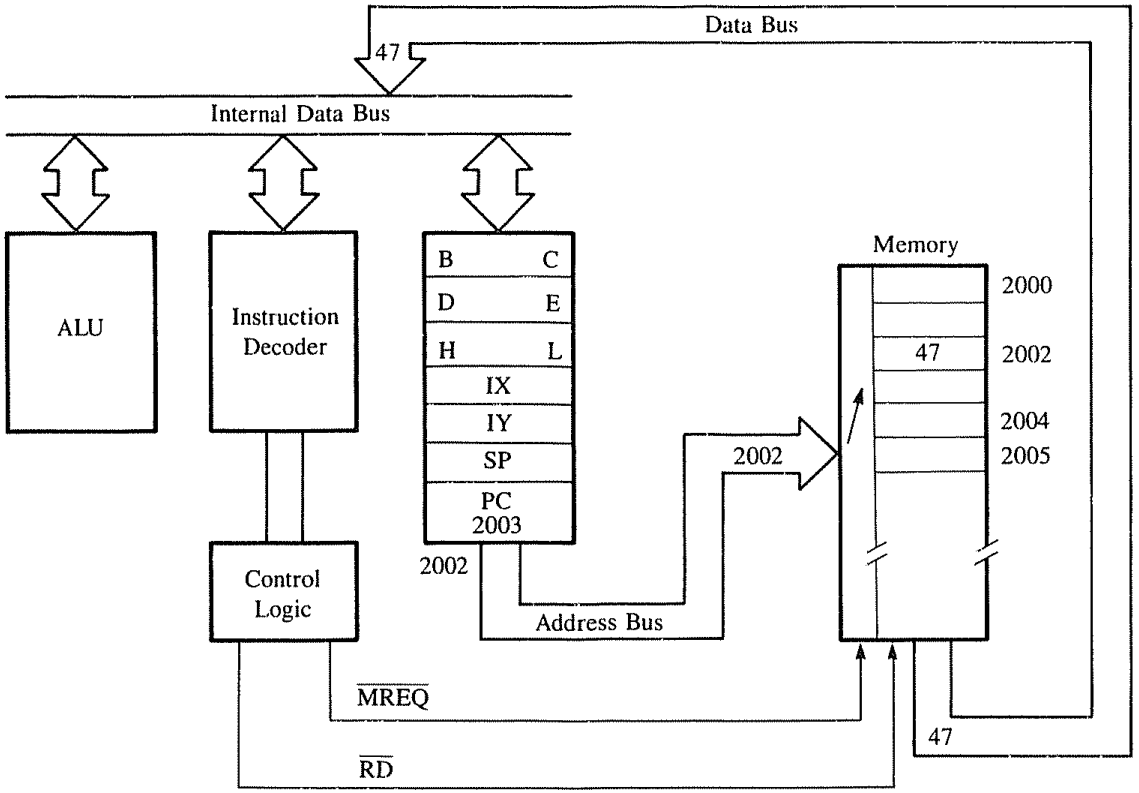


FIGURE 3.5
Z80 Memory Read Operation

سیکل ماشینی خواندن از حافظه (Memory Read)

- دستور دوبایتی LD A, 9FH را در نظر بگیرید. این دستور دارای دو سیکل ماشینی می باشد.
- سیکل ماشین فراخوانی opcode و سیکل ماشین خواندن ازحافظه .

Address	Machine Code			Instruction	Comment
2000_H	0 0 1 1 1 1 1 0	→	3E	LD A, 9FH	;Load 9FH in the accumulator
2001_H	1 0 0 1 1 1 1 1	→	9F		

سیکل ماشینی نوشتن در حافظه (Memory Write Cycle)

The HL register holds the address 2350_H, and the accumulator has the data byte $9F_H$. The instruction code 0111 0111 (77_H) is stored in memory location 2003_H. When this code is executed, it stores the contents of the accumulator in the memory location indicated by the address in the HL register. Illustrate the bus contents and timings as this instruction is being executed.

Instruction: LD (HL), A ;Copy contents of the accumulator into memory location, the address of which is stored in HL register.

This is a one-byte instruction with two machine cycles: Opcode Fetch and Memory Write. In the first machine cycle, the Z80 fetches the code (77_H), and in the second machine cycle, it copies the byte $9F_H$ from the accumulator into the memory location 2350_H. The

مفاهیم مهم:

۱- در هر سیکل دستورالعمل، اولین عمل(سیکل ماشینی) همیشه فرخوانی آپکود (Opcode Fetch) می باشد و با صفر شدن سیگنال M1’ مشخص می شود. این سیکل بین ۴ تا ۶ پریود (T) را بخود اختصاص می دهد.

۲- سیکل خواندن از حافظه بسیار شبیه به سیکل فراخوانی آپکود می باشد. هر دواز سیگنال های کنترلی MREQ’ و RD’ استفاده می کنند و محتویات حافظه را می خوانند. اگر چه در سیکل فراخوانی آپکود، Opcode خوانده میشود و در سیکل خواندن از حافظه، داده یا آدرس ۸ بیتی خوانده میشود. تفاوت این دو سیکل از طریق سیگنال M1’ مشخص می شود.

۳- سیگنال های کنترلی MREQ’ و RD’ هردو برای خواندن از حافظه لازم می باشند.

۴- در سیکل نوشتن در حافظه، Z80 با استفاده از سیگنال های کنترلی MREQ’ و WR’ داده ایی را در حافظه می نویسد.

۵- در سیکل خواندن از حافظه Z80 سیگنال های MREQ’ , RD’ را اعمال تا حافظه را فعال و محل حافظه آدرس شده محتوی خود را برروی گذرگاه داده قرار می دهد. در صورتی که در سیکل نوشتن در حافظه ، Z80 سیگنال MREQ’ را اعمال و داده را برروی گذرگاه داده قرار می دهد سپس سیگنال WR’ را اعمال تا داده در داخل محل حافظه آدرس شده نوشته شود.

۶- معمولا سیکل های خواندن و نوشتن شامل ۳ و در بعضی مواقع ۴ پریود (T) می باشند. ضمنا سیکل های خواندن و نوشتن هرگز بطور همزمان اعمال نمی شوند.

برنامه نویسی زبان اسمبلی Z80

- برای نوشتن یک برنامه به زبان اسمبلی Z80 ما باید از قابلیت های ریزپردازنده و مجموعه دستورالعمل های آن آشنا بشیم.
- مجموعه دستورالعمل های یک ریزپردازنده توانایی عملکرد آن، قدرت دستکاری داده هایش و سهولت برنامه نویسی آن را نشان میدهد.
- ریزپردازنده Z80 دارای ۱۵۸ نوع دستورالعمل می باشد.
- هر دستورالعمل از دو قسمت تشکیل می شود. یک قسمت نوع عملی که باید انجام شود را مشخص می سازد (Opcode) .
- و قسمت دوم داده ایی که عمل باید برروی آن انجام شود را مشخص می سازد(Operand) .

قالب دستور

- یک دستور فرمانی است به ریزپردازنده جهت انجام یک کار خاص.
- اندازه دستورات ریزپردازنده Z80 از یک بایت تا چهار بایت متفاوت میباشد.
- بنابراین، تعداد محل های حافظه مورد نیاز برای ذخیره نمودن این دستورات در حافظه نیز متفاوت می باشند.
- برای مثال برای نوشتن یک دستور سه بایتی در حافظه به سه محل حافظه نیاز می باشد.

- اکثر Opcode ها در یک بایت مشخص میگردند. البته Opcode های خاصی نیز هستند که به دو بایت نیازمندند.

- Operand یا داده ها می توانند به صورت های زیر نشان داده شوند.

- داده ۸ بیتی
- داده ۱۶ بیتی
- ثبات ها
- جفت ثبات ها
- آدرس حافظه

- مجموعه دستورالعمل های Z80 بر اساس اندازه هر دستور به چهار گروه یک بایتی، ۲ بایتی، ۳ بایتی و ۴ بایتی تقسیم می شوند.

- از آنجایی که ریزپردازنده Z80 یک پردازنده ۸ بیتی می باشد، واژه بایت و کلمه بجای یکدیگر استفاده می شوند.

دستورات یک بایتی

در یک دستور یک بایتی Opcode و Operand در یک بایت نمایش داده می شوند.

Task	Opcode	Operand	Binary Code
Copy the contents of register B into the accumulator A.	LD	A, B	01 111 000 (78H)
Add the contents of register B to the contents of A.	ADD	A, B	10000 000 (80H)

دستورات دو بایتی

2-BYTE INSTRUCTIONS

In a 2-byte instruction, the first byte specifies the opcode and the second byte specifies the operand (with exceptions of some Z80 two-opcode instructions).

Task	Opcode	Operand	Binary Code	
Load register B with the hexadecimal number 32. (Opcode for LD B is 06H)	LD	B, 32H*	0000 0110 (06H) 0011 0010 (32H)	Byte 1 Byte 2

دستورات سه بایتی

3-BYTE INSTUCTIONS

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address or data in a reversed order: low-order byte followed by the high-order byte. For example:

Task	Opcode	Operand	Binary Code	
Copy data from memory address 2080_H into the accumulator.	LD	A, (2080H)	0011 1010 (3AH)	Byte 1
			1000 0000 (80H)	Byte 2
			0010 0000 (20H)	Byte 3

دستورات چهار بایتی

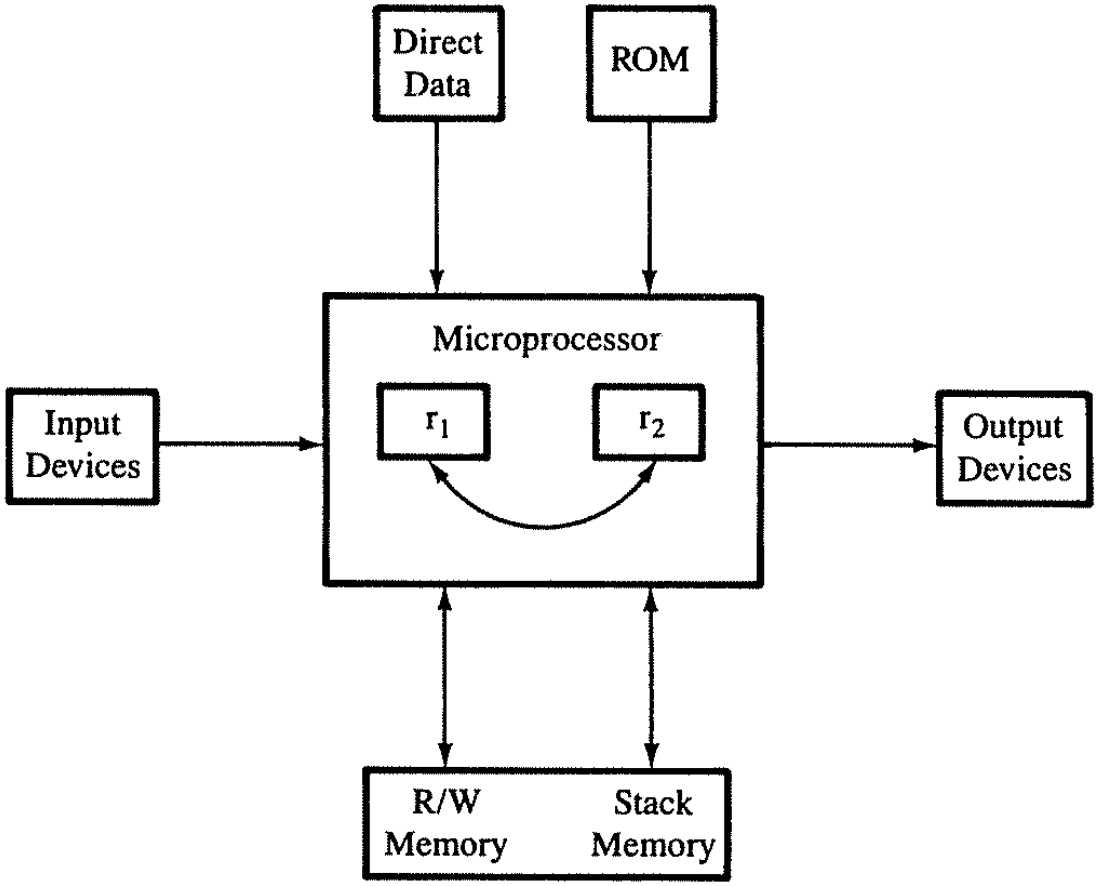
دستورات چهاربایتیZ80 عموما مربوط به ثبات های ایندکس می باشد.

Task	Opcode	Operand	Binary Code	
Load index register IX with 16-bit address 2000_H.	LD	IX, 2000H	1101 1101 (DDH)	Byte 1
			0010 0001 (21H)	Byte 2
			0000 0000 (00H)	Byte 3
			0010 0000 (20H)	Byte 4

مجموعه دستورالعمل های Z80 در ۶ گروه دسته بندی می گردد.

۱- کپی کردن داده ها یا بارگذاری (انتقال داده)
۲- عملیات ریاضی
۳- عملیات منطقی
۴- عملیات بر روی بیتها
۵- عملیات شاخه شدن یا پرش
۶- عملیات کنترل ماشین

Z80 دارای تعداد زیادی دستور می باشد که داده ایی را ازیک محل (Source) در محل دیگری (Destination) کپی می نماید، بدون اینکه محتوی منبع تغییر کند.



Data Copy Operations	Examples
1. From one register into another register.	Copy the contents of register B into the accumulator. LD A, B; LD means Load
2. (a) Specific data byte into a register or a memory location.	Load register B with the hexadecimal number 32. LD B, 32H
(b) Specific 16-bit data into a register pair.	Load register pair HL with hexadecimal number 2050. LD HL, 2050H
3. From a memory location into a register or vice versa.	Copy data from memory location 2080_H into the accumulator. LD A, (2080H)
4. (a) From an input port into the accumulator.	Read data from input port 01_H and copy into the accumulator. IN A, (01H)
(b) From the accumulator into an output port.	Write (send) the contents of the accumulator into port 07_H. OUT (07H), A
5. From microprocessor registers into stack memory locations and vice versa.	Copy the contents of register pair BC into defined stack memory locations. PUSH BC
6. Exchange contents between registers. (This is a slightly different operation from data copy; this is a data exchange.)	Exchange the contents of general purpose registers (BC, DE, HL) with alternate registers. EXX

1. Data Copy (Load) Instructions

Mnemonics	Bytes	Tasks
Data (8 bits and 16 bits) copy or load in registers		
LD r_d, r_s	1	Copy data from source register r_s into destination register r_d.
LD r, 8-bit	2	Load 8-bit into a register.
LD rp, 16-bit	3	Load 16-bit into register pair.
LD rx, 16-bit	4	Load 16-bit data into index register.

Data copy between registers and memory

LD A, (16-bit)	3	Load accumulator from memory; the address is specified by 16-bit operand.
LD (16-bit), A	3	Load memory from accumulator; the memory address is specified by 16-bit operand.
LD A, (rp)	1	Load accumulator from memory; the memory address is specified by contents of register pair.
LD (rp), A	1	Load memory from accumulator; the memory address is given by the contents of register pair.
LD r, (HL)	1	Load register from memory; the address is specified by 16-bit contents in HL.
LD (HL), r	1	Load memory from register; the address is specified by 16-bit contents in HL.
LD r, (rx + d)	3	Copy memory contents into register r; the memory address is obtained by adding the contents of index register and the displacement byte d.
LD (rx + d), r	3	Copy register contents into memory address shown by index register and the displacement (rx + d)

عملیات ریاضی

- مجموعه دستورالعمل های Z80 دارای چهار نوع عملی ریاضی میباشد:
- جمع
- تفریق
- Increment/Decrement
- مکمل یک و مکمل دو

- **Addition.** Any 8-bit number, or the contents of a register, or the contents of a memory location can be added to the contents of the accumulator. The result of the addition is stored in the accumulator, and the flags are affected by the result. No two other 8-bit registers can be added directly; for example, the contents of register B cannot be added directly to the contents of register C.

  Examples: Add the contents of register B to the contents of the accumulator. → ADD A, B

  Add the byte 97_H to the contents of the accumulator. → ADD A, 97H

- **Subtraction.** Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator. The subtraction is performed in 2's complement, and the result is stored in the accumulator. The result modifies the flags, and if the result is negative, it is expressed in 2's complement. The following mnemonics indicate that the accumulator is implicitly assumed as one of the operands.

  Examples: Subtract the contents of register C from the contents of the accumulator. → SUB C

  Subtract the byte 47_H from the contents of the accumulator. → SUB 47H

- **Increment/Decrement.** The 8-bit contents of a register (including the accumulator) or a memory location can be incremented or decremented by 1. Similarly, the 16-bit contents of a register pair (such as HL) can be incremented or decremented by 1. Unlike Add and Subtract, these operations can be performed in any of the registers. The instructions related to 8-bit contents affect flags (except Carry); on the other hand, instructions related to 16-bit contents do not affect any flags.

  Examples: Increment the contents of register B. → INC B
  Decrement the contents of register pair BC. → DEC BC

- **1's and 2's Complement.** The contents of the accumulator can be complemented (1's or 2's complement), and the result is stored in the accumulator. Some flags are affected by the result. These instructions assume that the operand is the accumulator.

  Examples: Complement the contents of the accumulator (this is equivalent to 1's complement). → CPL
  Subtract the contents of the accumulator from zero (this is equivalent to 2's complement). → NEG

2. Arithmetic Instructions*

ADD A, r	1	Add register contents to accumulator.
ADD A, 8-bit	2	Add 8-bit data to accumulator.
ADD A, (HL)	1	Add memory contents to accumulator; the memory address is specified by the contents in HL.
SUB r	1	Subtract contents of register from accumulator.
SUB 8-bit	2	Subtract 8-bit data from accumulator.
SUB (HL)	1	Subtract memory contents from accumulator; the memory address is specified by the contents of HL.
INC r	1	Increment the contents of a register.
INC (HL)	1	Increment the contents of memory; the memory address is specified by the contents of HL.
INC rp	1	Increment 16-bit contents in a register pair.
DEC r	1	Decrement the contents of a register.
DEC (HL)	1	Decrement the contents of memory; the memory address is specified by the contents of HL.
DEC rp	1	Decrement 16-bit contents in a register pair.

*Instructions used for 16-bit addition and subtraction are not shown here.

عملیات منطقی

LOGIC OPERATIONS

The instructions related to logic operations can be divided into three groups: logic functions (AND, OR, etc), bit rotations or shifts, and comparisons (less than, greater than, and equal to) of data bytes.

- **Logic Functions.** Any 8-bit number, the contents of a register, or the contents of a memory location can be ANDed, ORed, or Exclusive ORed with the contents of the accumulator. The result is stored in the accumulator, and the flags are affected by the result.

  Examples: Logically AND the contents of register B with the contents of the accumulator. → AND B

  Exclusive OR the contents of register B with the contents of the accumulator. → XOR B

3. Logic Instructions*

AND r	1	Logically AND the contents of a register with the accumulator.
AND 8-bit	2	Logically AND 8-bit data with accumulator.
AND (HL)	1	Logically AND the contents of memory with accumulator; the memory address is specified by the contents of HL.
CP r	1	Compare the contents of register with accumulator for less than, equal to, or greater than.
CP 8-bit	1	Compare 8-bit data with accumulator for less than, equal to, or greater than.
CP (HL)	1	Compare the contents of memory with accumulator for less than, equal to, or greater than. The memory address is specified by the contents of HL.

- **Shift and Rotate.** Each bit in the accumulator, in the registers, or in memory can be shifted either left or right by one position.

  Examples: Rotate the contents of the accumulator → RRA
  right through Carry flag.
  Rotate left the contents of register B. → RLC B

- **Compare.** Any 8-bit number, the contents of a register, or memory can be compared for equality, greater than, or less than with the contents of the accumulator. The result of the comparison is indicated by appropriate flags.

  Examples: Compare the contents of register B with the → CP B
  contents of the accumulator.
  Compare the data byte 97_H with the contents → CP 97H
  of the accumulator.

4. Bit Rotation

RLCA	1	Rotate each bit in the accumulator to the left position.
RLA	1	Rotate each bit in the accumulator including the carry C to the left position.
RRCA	1	Rotate each bit in the accumulator to the right position.
RRA	1	Rotate each bit in the accumulator including the carry C to the right position.

7. Bit Rotation*

RLC r	2	Rotate each bit in register r to the left.
RL r	2	Rotate each bit in register r to the left, including Carry flag.
SLA r	2	Shift each bit in register r to the left.

عملیات برروی بیت ها (Bit Manipulation)

The bit manipulation instructions can be classified into two groups: bit test and bit set/reset.

- **Bit Test**—Any one of the eight bits in a register, accumulator, or memory can be verified as 0 or 1, and the Z flag will be modified accordingly.

  Example: Check bit D_7 in register B. → BIT 7, B

- **Bit Set/Reset**—Any one of the eight bits in a register, accumulator, or memory can be set or reset.

  Examples: Set bit D_5 in the accumulator. → SET 5, A
  Reset bit D_2 in register B. → RES 2, B

8. Bit Manipulation†

BIT b, r	2	Test bit b in register r, affecting the Z flag.
SET b, r	2	Set bit b in register r. (“b” represents bit position 0 to 7)
RES b, r	2	Reset bit b in register r.

(Branching Operation) عملیات شاخه شدن یا پرش

- **Jump.** The sequence of program execution can be altered either conditionally or unconditionally. When a conditional Jump instruction is used, the microprocessor checks the specified flag, and if the condition is true, the execution sequence is altered; otherwise, the next instruction is executed. The destination location to which the program should be directed can be specified directly or relative to the contents of the program counter. These instructions are critical to the decision making process in programming.

  Examples: After an operation (such as an addition), if CY flag is set, jump to location 2050_H. → JP C, 2050H
  If Zero flag is not set, jump forward by 15 locations. → JR NZ, 0FH

- **Call/Return.** These instructions change the sequence of a program by calling a subroutine or returning from a subroutine. The conditional Call and Return instructions check for appropriate flags.

  Examples: Go to subroutine located at 2050_H. → CALL 2050H
  Go to Subroutine located at 2070_H if Z flag is set. → CALL Z, 2070H

- **Restart.** These instructions are used to change the program sequence to one of eight restart locations on memory page 00. The instructions are generally used with interrupts.

  Example: Call location 0028_H. → RST 28H

5. Branch Instructions†

JP 16-bit	3	Change the program sequence (Jump) to memory location specified by the 16-bit address.
JP Z, 16-bit	3	Change the program sequence (Jump) to memory location specified by the 16-bit address if the Zero (Z) flag is set.
JP NZ, 16-bit	3	Change the program sequence (Jump) to memory location specified by the 16-bit address if the Zero (Z) flag is reset.
JP C, 16-bit	3	Change the program sequence (Jump) to memory location specified by the 16-bit address if the Carry (C) flag is set.
JP NC, 16-bit	3	Change the program sequence (Jump) to memory location specified by the 16-bit address if the Carry (C) flag is reset.
CALL 16-bit	3	Change the program sequence to the location of the subroutine.
RET	1	Return to the calling program after completing the subroutine sequence.

عملیات کنترل ماشین

MACHINE CONTROL OPERATIONS

These instructions control microprocessor operations such as Halt and Interrupt.

Examples: Suspend execution of instruction. → HALT
Disable interrupts by resetting the → DI
Interrupt Enable flip-flops.

6. Machine Control Instructions

HALT	1	Suspend execution and wait.
NOP	1	Do not perform any operation.

روش های آدرس دهی Addressing Modes

- دستورالعمل ها بروی اطلاعاتی که در ثباتهای داخلی یا حافظه های جانبی و یا تراشه های ورودی و خروجی میباشد عملیاتی را انجام میدهند.

- هر دستورالعمل علاوه بر تعیین نوع کار مشخص میکند که دستورالعمل مربوط به ثباتهای درون ریزپردازنده یا تراشه های حافظه و یا تراشه های ورودی و خروجی است. تعیین این موضوع را آدرس دهی گویند.

- آدرس دهی به روشی گفته میشود که ریزپردازنده محلی که عملیات باید صورت گیرد را در موقع عملیات تولید میکند.

- روش آدرس دهی یک راه مشخص نمودن یک Opreand یا اشاره کردن به یک محل حافظه است.
- ریزپردازنده Z80 دارای ۱۰ روش آدرس دهی می باشد.

Z80 Addressing Modes	Explanation	Example
1. Immediate :	The byte following the opcode is the operand. This mode is used to load 8-bit data into a register.	
	Load 97_H into register B	LD B, 97H
2. Immediate Extended :	The two bytes following the opcode are the operands. This mode is used to load 16-bit data or address into a register pair.	
	Load 8045_H into register pair BC	LD BC, 8045H
3. Register :	The operand register is included as a part of the opcode. This mode is used to copy data from one Z80 register into another register.	
	Copy data from register A into B	LD B, A
4. Implied :	This refers to operations in which the opcode implies one or more Z80 registers as containing the operands. For example, instructions for logic operations imply that the accumulator is one of the operands and that the result is stored in the accumulator.	
	Logically AND register B with A	AND B
5. Register Indirect :	This mode is used to copy data between the MPU and memory; the 16-bit contents in a register pair are used as a memory pointer.	
	Copy the contents of memory location 2060_H into register B. Register HL contains the address 2060_H.	LD, B, (HL)
6. Extended :	The two bytes following the opcode specify the jump location.	
	Jump to location 2080_H.	JP 2080H.
7. Relative :	In this mode, the second byte specifies the displacement value in a signed 2's complement for a jump location.	
	Jump forward 20 locations from the address of the next instruction.	JR 14H
8. Indexed :	In this mode, the byte following the opcode specifies a displacement value that is added to one of the index registers to form a memory pointer.	
	The index register IX contains 2060_H; increment the contents of memory location 2070_H.	INC (IX + 10H)
9. Bit :	This mode is used for bit operation (manipulation). In this mode, instruction specifies a bit from a register or a memory location using one of the three addressing modes (register, register indirect, or indexed).	
	Set bit D_7 in register B.	SET 7, B
10. Page Zero :	The instruction set includes eight restart (one-byte call) instructions on memory page zero. In this mode, the memory location can be specified by using the low-order byte, and the high-order byte is assumed to be 00_H.	
	Call restart memory location 0028_H.	RST 28H

مثال:برنامه ایی بنویسد که دو عدد هگز 32H و A2H را به ترتیب در ثبات های B وC قرار داده و حاصل جمع این دوعدد را از طریق پورت شماره ۱ نشان دهد.

```
LD B, 32H        ;Load register B with 32H.
LD C, A2H        ;Load register C with A2H.
LD A, C          ;Copy contents of C into accumulator to perform addi-
                  tion. B and C cannot be added directly.
ADD A, B         ;Add two bytes and save the sum in A.
OUT (01H), A     ;Display accumulator contents at port 01H.
HALT             ;End
```

Mnemonics	Hex Code	
LD B, 32H	06 32	2-byte instruction
LD C, A2H	0E A2	2-byte instruction
LD A, C	79	1-byte instruction
ADD A, B	80	1-byte instruction
OUT (01H), A	D3 01	2-byte instruction
HALT	76	1-byte instruction

Mnemonics	Hex Code	Memory Contents	Memory Address
LD B, 32H	06	0 0 0 0 0 1 1 0	2000
	32	0 0 1 1 0 0 1 0	2001
LD C, A2H	0E	0 0 0 0 1 1 1 0	2002
	A2	1 0 1 0 0 0 1 0	2003
LD A, C	79	0 1 1 1 1 0 0 1	2004
ADD A, B	80	1 0 0 0 0 0 0 0	2005
OUT (01H), A	D3	1 1 0 1 0 0 1 1	2006
	01	0 0 0 0 0 0 0 1	2007
HALT	76	0 1 1 1 0 1 1 0	2008

تمرین ۱ : برنامه زیر را به زبان ماشین تبدیل کنید.

```
START:   LD B, 32H          ;Load B with first data byte
         LD C, 0A2H         ;Load C with second data byte
         LD A, C            ;Copy (C) into A for addition
         ADD A, B           ;Add two bytes
         JP NC, DSPLAY      ;If sum < FFH, display sum at PORT7
         LD A, 01H          ;If sum > FFH, load 01 to display
                            ; as over load
DSPLAY:  OUT (PORT7), A     ;Display result at PORT7
         HALT
         END
```

مثال: برنامه زیر را تحلیل نمایید.

Instructions			Register Contents					Flags		
Mnemonics			*A*	*B*	*C*	*H*	*L*	*S*	*Z*	*CY*
1. LD BC, F268H			X	F2	68	X	X	No change		
2. LD HL, 2065H			X	↓	↓	20	65	↓	↓	↓
3. LD (HL), A2H	A2	2065	X	↓	↓	↓	↓	↓	↓	↓
4. LD A, B			F2	↓	↓	↓	↓	↓	↓	↓
5. SUB C (F2 − 68) →			8A	↓	↓	↓	↓	1	0	0
6. CPL (Invert 8A) →			75	↓	↓	↓	↓	No change		
7. ADD A, (HL) (75 + A2)			17	↓	↓	↓	↓	0	0	1
8. INC HL			↓	↓	↓	20	66	No change		
9. LD (HL), A	17	2066	↓	↓	↓	↓	↓	↓	↓	↓
HALT			17	F2	68	20	66	0	0	1

مثال : ثبات های شاخص IX و IY را به ترتیب برای اشاره به محل های حافظه 2050H و 2150H در نظر بگیرید. و با استفاده از این ثباتها عد 32H را درمحل حافظه 2090H و عدد 97H را در محل حافظـه 2110H قـرار و سـپس این دو عدد را جمع و حاصل را در انباره قرار دهید.

```
LD  IX ,2050H
LD  IY ,2150H
LD(IX+40H), 32H
LD(IY+C0H), 97H
LD A,(IX+40H)
ADD A ,(IY+C0H)
HALT
```

مثال: ۱۰۰ بایت داده که بطور متوالی در حافظه ذخیره شده اند را به ترتیب به بخش دیگری از حافظه انتقال دهید.

```
Label    Mnemonics          Comments

START:   LD HL, SOURCE      ;Set up HL as pointer for Source
                              memory
         LD DE, OUTBUF      ;Set up DE as pointer for Output
                              Buffer memory
         LD B, 64H          ;Set register B to count 100 bytes

NEXT:    LD A, (HL)         ;Get byte from Source
         LD (DE), A         ;Store byte in Output Buffer

         INC HL             ;Point to next Source location
         INC DE             ;Point to next Output Buffer
                              location
         DEC B              ;Decrement count

         JP NZ, NEXT        ;If counter is not zero, go back
                              to get next byte

          LD A, 01H         ;Load display indicator
          OUT (PORT0), A    ;Display end of data transfer
          HALT              ;End of program
```

تمرین ۲

- وضعیت Stack و SP را بعد از اجرای برنامه نشان دهید.

```
        LD SP, 219AH
        LD B , 03H
        XOR A
        LD H, A
        LD L, A
LOOP: PUSH HL
        INC L
        DEC B
       JP NZ , LOOP
        HALT
```

تمرین ۳ : برنامه زیر را درنظر گرفته و به پرسش های صفحه بعد پاسخ دهید.

3. Read the following program and answer the questions

No.	Instructions
1.	LD SP, 84F9H
2.	LD HL, 8138H
3.	LD BC, 0001H
4.	LD DE, 235AH
5.	LD A, D
6.	OR A
7.	PUSH HL
8.	PUSH AF
9.	PUSH BC
	↓
20.	POP AF

۱- محتوی sp بعد از اجرای اولین دستور چیست؟

۲- محتوی محل های حافظه 84F8H و 84F7H بعد از اجرای دستور ۷ (PUSH HL) را پیدا کنید.

۳- محتویات انباره و ثبات پرچم را بعد از اجرای دستور OR A بدست آورید.

۴- محتوی SP و محتویات دو محل بالای Stack بعد از اجرای دستور شماره ۹ را نشان دهید.

۵- بعد از اجرای دستور ۲۰ محتویات پرچم های S,Z,C را مشخص کنید.

حل تمرین۱

2000	LD B,32H	06	32		7T
2002	LD C,A2H	0E	A2		7T
2004	LD A,C	79			4T
2005	ADD A,B	80			4T
2006	JP NC,200BH	D2	0B	20	10T
2009	LD A, 01H	3E	01		7T
200B	OUT(7H), A	D3	07		11T
200D	HALT	76			4T

					54T

حل تمرین ۲

SP	STACK
2194	02
2195	00
2196	01
2197	00
2198	00
2199	00
219A	