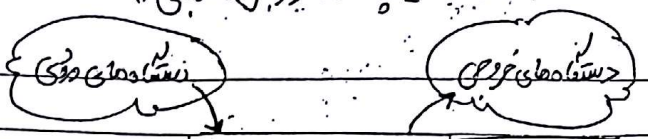
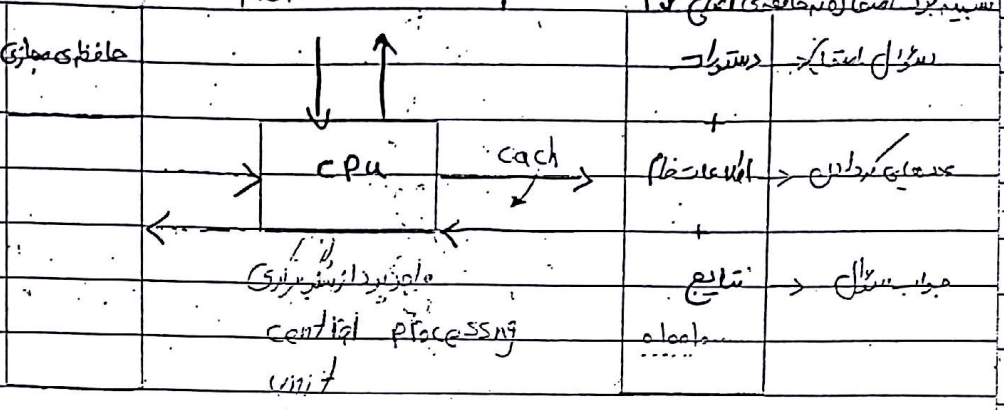
ریز پردازنده و زبان اسمبلی ۱

۹۵/۷/۴ جلسه ۱ ۳۱۲



تشبیه برای اصطلاحات حافظه‌ی اصلی



حافظه‌ی اصلی (اولیه) — حافظه‌ی جانبی (ثانویه) secondary memory

main memory (سرعت بالا و گرانقیمت) — virtual memory (سرعت پایین‌تر)

همه‌ی سیستم‌های کامپیوتر از واحدی تشکیل شده‌اند به نام cpu. کار cpu پردازش اطلاعات است. پردازش یعنی روی داده‌ها پردازش انجام دهیم (عملیات ریاضی مثل جمع، ضرب، تقسیم) عملیات منطقی و حسابی. پردازش حافظه‌ی اصلی واحدی است که می‌تواند همه‌ی اطلاعات خود را در خودش نگهداری کند. سه حال نگهداری می‌توانند موقت (یعنی تا زمانی که برق وصل باشد) و دائم است چون برق نباشد اطلاعات ذخیره می‌کند. Hard Disk دائم) و Ram کامپیوتر موقت)

هر حافظه‌ی اصلی ما باید از قبل دستورات و اطلاعات خام را داخلش قرار بدهیم در داخل حافظه‌ی اصلی جواب ما و نتایج در داخل آن ذخیره می‌شود (یکی از کاربردهای حافظه‌ی اصلی)

* حافظه‌های اصلی عموماً اکثر آن‌ها به شکل موقت هستند یعنی تا زمانی که برق به آن‌ها وصل است محتویات داخل آن‌ها مانند اگر برق قطع شود از بین می‌روند. البته انواع خاصی هستند که حافظه‌ی آن‌ها دائم است.

کامپیوتر ما برای این مراحلی شده است که بتواند از اطلاعات را با دقت و سرعت بالا پردازش کند.

برای اینکه با قطع برق اطلاعات از بین نرود یک حافظه‌ی جانبی مراحل کردند.

حافظه‌های جانبی دیتا فریم این خاصیت را دارند که دائمی هستند. بدون نیاز به جریان برق می‌توانند محتویات داخل خود را حفظ کنند. ما برای اینکه بتوانیم اوامر و دستورات و اطلاعات خام که در حافظه‌ی اصلی هستند ممکن است بعداً: به آن‌ها احتیاج داشته باشیم با قطع برق از بین نرود آن‌ها را به حافظه‌ی جانبی انتقال می‌دهیم. وقتی بخواهیم از آن‌ها استفاده کنیم آن‌ها را از حافظه‌ی جانبی به حافظه‌ی اصلی آورده و از آن‌ها استفاده می‌کنیم.

یک برنامه یک مجموعه‌ای از دستورات است که وقتی پشت سر هم اجرا شود یک کاری برای ما انجام می‌شود. برنامه‌ای که معادل با بیشتر صحبت می‌کنیم در حافظه‌ی جانبی ذخیره می‌شود و با کپی کردن از حافظه‌ی جانبی وارد حافظه‌ی اصلی شده و از آن استفاده می‌کنیم.

Scanned by CamScanner

دهمین CPU از حافظه‌ی اصلی دستور می‌گیرد

حافظه‌ی جانبی از حافظه‌ی اصلی ارزان‌تر هستند و حجم بیشتری از اطلاعات را ذخیره می‌کنند

سرعت حافظه‌های اصلی از حافظه‌ی جانبی بیشتر است

به خاطر اینکه سرعت جانبی نسبت به حافظه‌های اصلی خیلی کمتر است بنابراین ناگزیر به استفاده حافظه‌ی اصلی بودیم اگر نمی‌توانستیم CPU را مستقیم به حافظه‌ی جانبی وصل کنیم البته تو سیستم‌های کامپیوتری فعلی ارتباط مستقیم CPU با حافظه‌ی جانبی وجود دارد. حافظه‌ی اصلی هم ظرفیت کم هم گران قیمت هستند خیلی وقت‌ها ظرفیت حافظه‌ی اصلی جوابگوی ما نیست ما می‌آییم قسمتی از حافظه‌ی مجازی را به عنوان حافظه‌ی اصلی استفاده می‌کنیم و CPU فکر می‌کند حافظه‌ی مجازی اطلاعاتی از حافظه‌ی اصلی است ما اطلاعات بیشتر حافظه‌ی مجازی هست که در حافظه‌ی اصلی پردازش نشده را در حافظه‌ی مجازی با ریخت پاشی استفاده کنیم

در سیستم کامپیوتر باید برآمی داشته باشیم تا دستورات و اطلاعات خام را وارد حافظه‌ی اصلی یا جانبی بکنیم و بتوان نتایج را به دنیای خارج ببریم (مثل نفر اشال تازه‌ای که حرف نزنیم نمی‌توان فهمید در مغزش می‌گذرد) واحد I/O یا ورودی و خروجی این واحد I/O این امکان را فراهم می‌کند تا ما بتوانیم اطلاعات و دستورات را داخل حافظه‌ی اصلی بکنیم و بالعکس نتایج را ببینیم (واحد I/O مثل چشم دهن و گوش انسان می‌ماند که فرامین که به ما داده می‌شود و از طریق نشانه‌ی به ما منتقل می‌شود یا چشم و نتایج را از طریق دهان یا نوشتاری خارج می‌کنیم)

اطلاعات و دستورات از راه‌های مختلف می‌توانند وارد یک کامپیوتر بشوند و از راه‌های مختلفی می‌توانند خارج بشوند که اصطلاحاً ما به کل این معادی ورودی و خروجی کامپیوترها پورت‌های کامپیوتر می‌گوییم یک کامپیوتر برای تعدادی پورت هست بعضی‌ها برای ورود هستند از طریق آن‌ها اطلاعات و دستورات می‌توانند وارد بشوند و بعضی از پورت‌ها برای خروج باشند و بعضی از پورت‌ها می‌توانند دوطرفه باشند ما به کل پورت‌های یک کامپیوتر واحد I/O می‌گوییم. واحد I/O با دستگاه‌های ورودی و خروجی ارتباط دارند.

دستگاه‌های ورودی و خروجی جزء یک سیستم کامپیوتری به حساب نمی‌آیند. کامپیوترها سیستم‌های دیجیتال هستند دیگر سیستم دیجیتال همه موجودیت‌ها (غیر فیزیکی) اطلاعات و دستورات به شکل صفر و یک هستند) در داخل حافظه‌ی کامپیوتر ما انبوهی از صفر و یک‌ها داریم که این صفر و یک‌ها می‌توانند نشان دهند دستورات و اطلاعات باشند. این صفر و یک‌ها در داخل CPU و حافظه‌ی اصلی و حافظه‌ی جانبی در حال تردد هستند. واحد I/O امکان انتقال صفر و یک‌ها را فراهم می‌کند.

دستگاه‌های خروجی صفر و یک‌ها را می‌گیرند به شکل قابل فهم ما نشان می‌دهند. مثلاً مانیتور صفر و یک‌ها را نشان می‌دهد

دستگاه‌های ورودی دستورات و اطلاعات را به شکل قابل فهم ما دریافت می‌کنند و به صفر و یک تبدیل می‌کنند. کیبورد و موس - اسکنر و ...

کار CPU پردازش اطلاعات است

سرعت حافظه‌ی اصلی هزاران برابر بیشتر از سرعت حافظه‌ی جانبی است

Scanned by CamScanner

جلسه‌ی دوم:

۹۵/۱۰/۱۷

صـ 3

واحدهای I/O کامپیوتر می‌تواند شامل تعدادی زیاد درگاه و یا پورت باشد. درگاه جایی است که اطلاعات می‌توانند داخل و خارج شوند.
معمولاً درگاه‌ها واحد I/O می‌نویسیم.

کامپیوتر یک سیستم دیجیتال است وقتی سیستم دیجیتال باشد همه چیز به شکل صفر و یک است یعنی همه‌ی چیزها دستورات، اطلاعات یا هر چیزی که می‌خواد وارد پردازش شود یا هر کاری که بخواد انجام بشه باید به صورت صفر و یک باشد.

صفر و یک‌های زیادی بین CPU و حافظه‌ی اصلی و حافظه‌های جانبی و I/O در حال رد و بدل است صفر و یک‌ها می‌توانند دستور باشند یا نشان دهنده‌ی اطلاعات خام یا اطلاعات پردازش شده‌ها باشند.

همه چیز در دنیای کامپیوتر به شکل صفر و یک است.

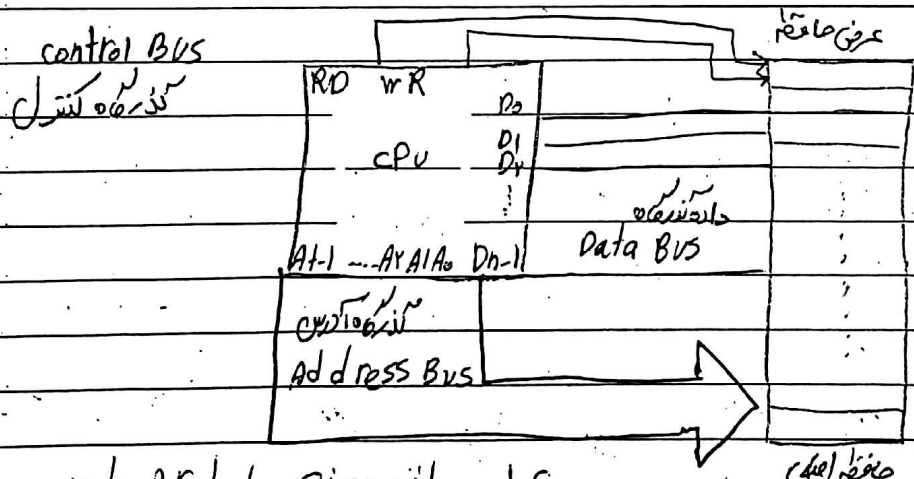
برای ما صفر و یک‌ها قابل فهم نیست پس برای رفع این مشکل آمده‌اند یک سری دستگاه‌های ورودی و خروجی در نظر گرفته‌اند.

وظیفه‌ی این دستگاه‌های ورودی این است که اطلاعات را به شکل قابل فهم ما می‌گیرند و به صفر و یک تبدیل می‌کنند مثلاً وقتی با کیبورد اسکلت می‌کنیم.

خروجی هم دستگاه‌های خروجی هستند که این کار را انجام می‌دهند یعنی صفر و یک‌ها را به شکل قابل فهم ما درمی‌آورد. مانیتور، اسپیکر

مینیمم سیستمی که یک کامپیوتر کند شامل CPU، حافظه‌ی اصلی و I/O است. سیستم حداقل.

خصوصی ارتباط CPU و حافظه‌ی اصلی یا پورت‌ها



Integrated Circuit = IC
مجتمع مدار

n = External data bus width
پهنای گذرگاه داده خارجی

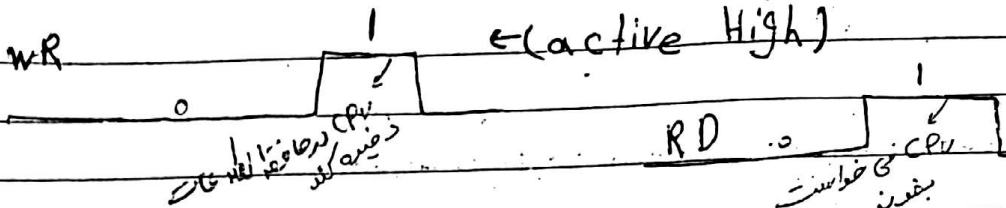
68 = 1000100 (2)

A15 A14 A0 A6 A5 A4 A3 A2 A1 A0
0 ... 0 0 0 0 0 0 0

Scanned by CamScanner

در حال تغییر وضعیت بوده

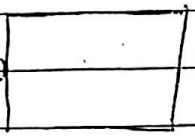
WR ← (active High)



CPU داده را در حافظه ذخیره کند

CPU می خواست بخواند

$\overline{WR}$



if address bus = t bit ⇒ حداکثر تعداد خانه‌های حافظه = 2^t

4 KB = 4 * 1024 = $2^2 * 2^{10} = 2^{12}$ ... $\log_2(2^{12}) = 12$ bit

$2^2 \times 2^{10} = 2^{12}$

بین CPU و حافظه‌ی اصلی ضرب‌ها و دال‌ها رد و بدل می‌شود.

روی هر CPU ما یک سری سیم و پایه داریم. CPU یک مدار مجتمع یا IC است.

معمولاً IC ها یک پایه‌هایی دارند که فقط سیم به IC های دیگه وصل میشه.

ما در داخل CPU هم یک سری پایه‌های خاص داریم که به حافظه‌ی اصلی وصل میشه که سیگنال‌هایی که می‌خواهند بین CPU و حافظه‌ی اصلی جابه‌جا شوند از این پایه‌ها عبور می‌کنند.

* به اولین مجموع پایه‌های داخل CPU که برای نقل و انتقال یا دستورات پیش‌بینی شده گذرگاه داده یا DataBus می‌گویند.

با این پایه‌ها که برای نقل و انتقال پیش‌بینی شده را $D_0, D_1, \dots, D_{n-1}$ نامگذاری می‌کنیم. ما n تا پایه در CPU داریم برای رد و بدل کردن صفر و یک‌ها بین CPU و حافظه‌ی اصلی پیش‌بینی شده.

* این n تا پایه گذرگاه داده یا DataBus می‌گویند که عدد n بیانگر پهنای گذرگاه داده است.

حافظه‌ی اصلی معمولاً یک سری سلول داره که این سلول‌ها می‌توانند یک بایت یا دو بایت را در خودشون ذخیره کنند که هر سلول یک شماره دارد که سلول‌ها از 0 تا $k-1$ تقسیم می‌شود.

* هر خونه حافظه یک آدرس دارد که همان شماره‌های آن‌ها است.

* اندازه‌ی هر سلول اون حجم اطلاعاتی که در هر خانه حافظه CPU می‌تواند در حافظه ذخیره کند یا بالعکس از حافظه بگیره عرض حافظه می‌گویند.

مثال: عرض یک حافظه 16 بیت باشد. هر خونه حافظه 16 بیت جا داره.

* عرض حافظه اول تعداد بیتی که هر خونه جا می‌شود.

روی CPU پایه‌های دیگری داریم که این پایه‌ها به حافظه‌ی اصلی وصل است که CPU همیشه قبل از اینکه بخواهد با حافظه ارتباط برقرار کند صفر و یک بگیره یا بفرسته شماره خونه‌ی

Scanned by CamScanner

مورد نظر یک آدرس آن را به صورت عدد مبنای ۲ یا به شکل معنوی یک روی این پایه ها قرار می دهد
منبع CPU اعلام می کند با این پایه ها را $A_0, A_1, \dots A_{n-1}$ می نامیم و به کل این پایه ها آن گذرگاه آدرس یا Address Bus می گوییم (CPU بتواند محل مورد نظر خودش را مشخص کند یا کدام خانه حافظه دارد)
مثال: CPU می خواهد اطلاعات را در خانه 48 ذخیره کند یا از آن بخواند
Address Bus 16 بیت است.

48 به مبنای ۲ = 00000000

می بینیم

* به + نهایی گذرگاه آدرس می گوییم

A15 A14 ..... A7 A6 A5 A4 A3 A2 A1 A0

0 0 ..... 0 1 0 0 0 0 1 0 0

CPU می داند اولین عددی که روی Address Bus قرار می گیرد که اولین عدد معادل مبنای ۲ عدد 48 است. خودش نباید بقیش با صفر پر می شود.
هر موقع CPU بخواهد محلی از حافظه را مورد دسترسی قرار بده شماره ی اول محل را به شکل مبنای ۲ روی Address Bus نشان می دهد.

* CPU با یک محل حافظه ۲ کار را انجام می دهد: یا از آن محل اطلاعات ذخیره کرده یا بنویسد یا از اون محل اطلاعات را بیرون دارد.
روی هر CPU دو تا پایه ی کنترلی داریم یک پایه بنام RD یک پایه دیگر WR
این دو پایه از مهمترین سیگنال های به حافظه ی اصلی وصل است. مقدار RD و WR زمانی که CPU با حافظه اصلی کاری ندارد صفر است. هر زمانی که CPU بخواهد در حافظه اطلاعات را ذخیره کند یا دستکاری WR خودش را فعال میکند و تغییر وضعیت میکند عکس آن صفر بود یک می کنیم

دو نوع داریم Active High ← وقتی فعال می شود در حالت عادی صفر است
Active Low ← در حالت عادی 1 و وقتی فعال شود صفر می شود.

معمولا آر سیگنال Active Low باشد یعنی بالای آن یک خط دارد $\overline{RD}$ $\overline{WR}$
اون هایی که خط روی آن ها نیست Active High است RD WR

آر سیگنال های Active Low معمولا یک دایره ی کوچک قبل از اسم سیگنال ما مشخص می کنیم

* ما به این دو تا سیگنال RD و WR به همراه سیگنال های دیگر گذرگه کنترل یا Control Bus می گوییم

Scanned by CamScanner

Data BUS دوطرفه است یعنی هم میتواند از CPU به حافظه اصلی برود هم برعکس

AddressBUS یک طرفه است همیشه اختیار دست CPU است. یعنی CPU خانه را انتخاب میکند

RD و WR یک طرفه است اختیار دست CPU است

Data BUS دوطرفه است یعنی زمان ها CPU میتواند و هم یعنی زمان ها حافظه میتواند را مشغول میکند

با تعداد بیت های آدرس میتوان آدرس میدهیم.

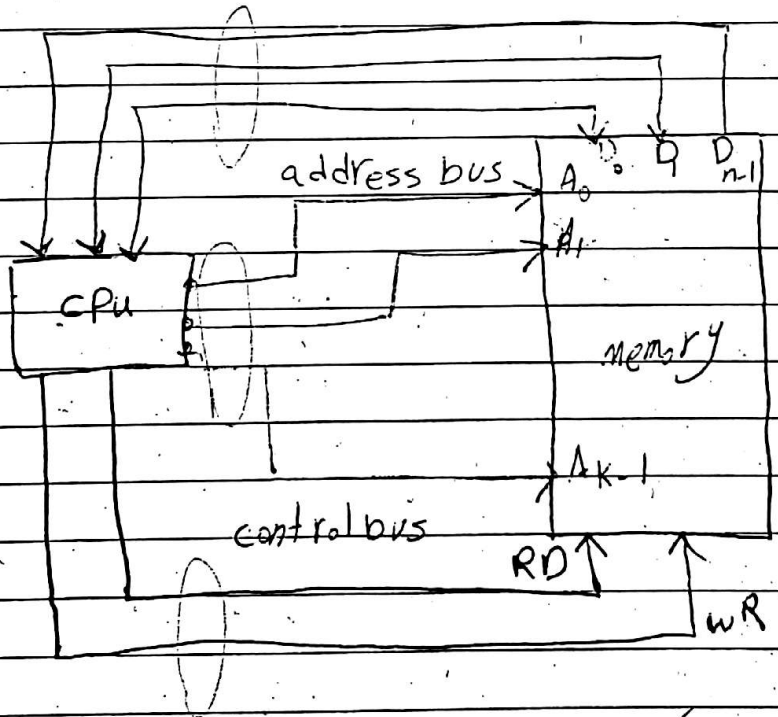
تعداد بیت AddressBus ها باشد حداکثر تعداد خانه های حافظه 2^n می شود

مثال حافظه ای داریم 4 kb آدرس BUS ، CPU باید چقدر باشد.

$4\ KB = 4 \times 1024 = 2^2 \times 2^{10} = 2^{12}$ خانه دارد $\log_2^{2^{12}} = 12\ bit$

AddressBus های CPU باید ۱۲ بیت باشد اگر کمتر باشد بالا از یکسری از خانه ها بالا استفاده می ماند.

آدرس بیت AddressBus با حافظه هایی داشته باشد با بتوان از n به ۲ ظرفیت استفاده کنیم



در اینجا می خواهیم ببینیم زمانی که CPU می خواهد در حافظه اطلاعاتی را بنویسد چه اتفاقی می افتد

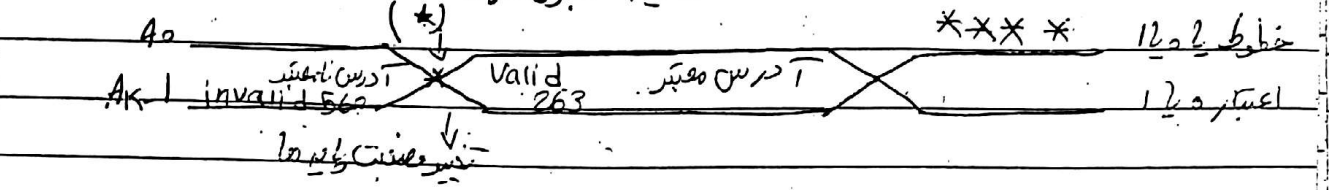
(CPU چه کاری را باید انجام دهد)

Scanned by CamScanner

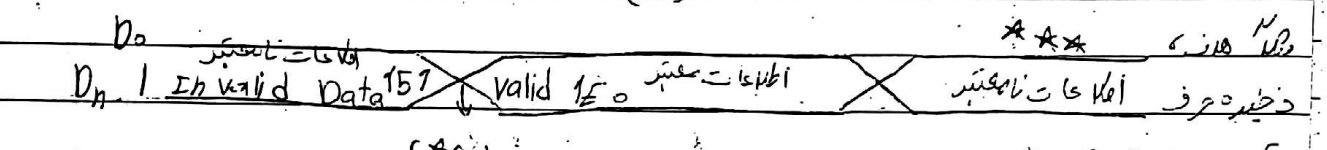
نمودار وضعیت سیگنال های CPU در عمل write: (سؤال امتحان)

(I)

حداینجا معتبری شوند

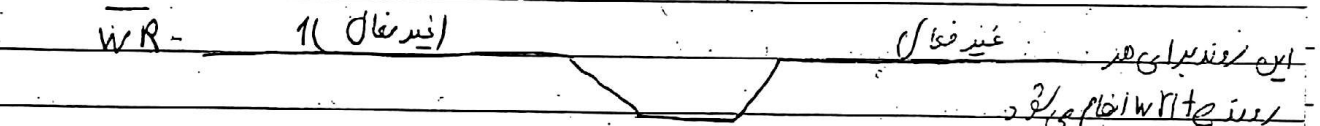


(II)



۳ درصد 263 حافظه بود

(III)



این روند برای هر عمل write انجام می‌شود

فعال سازی خط write داده‌ها را در حافظه می‌نویسد active low

(I) این شکل وضعیت address Bus در خطوط A_0 تا A_{k-1} را نشان می‌دهد.

(II) این شکل وضعیت data Bus در خطوط D_0 تا D_{n-1} را نشان می‌دهد

(III) این شکل وضعیت خطوط RD و WR را نشان می‌دهد.

ابتدای شکل نشان می‌دهد که عمل write را می‌خواهد انجام دهد.

CPU نوعی مدار دیجیتال است که با پیام‌ها یا دستورها کار می‌کند. این دستورها روی پایه‌های خود دارد.

یا مقدار سیگنال معتبر (valid) یا مقدار بی‌اعتبار (invalid) است. حالا وقتی پایه‌های valid و invalid چنین حالتی دارند، ما وقتی برق را به سیستم وصل کنیم، به طور تصادفی بعضی از پایه‌ها مقدار 1 و بعضی 0 هستند

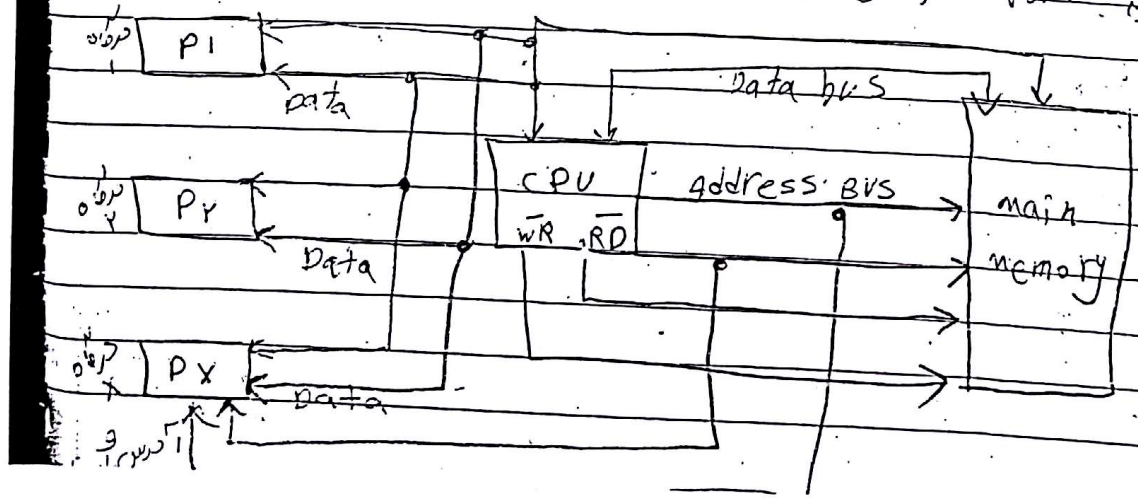
در قسمت اول شکل (I) یک سری پایه‌ها وجود دارد که نامعتبر است.

(در قسمت اول شکل (II) و (III) هم همین‌طور)

مقداری که خود CPU روی خط قرار می‌دهد معتبر است.

Scanned by CamScanner

در این شکل CPU می خواهد بخواند.
ما شروع write یا بیت آدرس valid می‌شود یعنی شما می‌خواهی مورد نظر بودی
address bus مشخص می‌شود و می‌گوید با چه خانه‌ای کار دارد. از طرفی می‌دانیم اختیار خطوط
data bus با CPU است یعنی یک لحظه‌هایی CPU می‌گوید کدام خطوط مال من و با آن‌ها یک
لحظه‌هایی حافظه می‌گوید. طبیعتاً در هر لحظه یکی از این دو تا مشخص می‌دهد. پس از این که آدرس
valid شد valid شد. data قبلاً valid بوده یعنی CPU یکسری چیزها را روی data bus مشخص
کرده که الان حاضر کردیم و آماده است.
CPU می‌آید و data bus خودش را در وضعیت float یا high impedance قرار می‌دهد.
یعنی CPU می‌گوید بچه‌ها من الان master هستم و دارم با یکی از شما صحبت می‌کنم و شما
را آدرس می‌دهم. master داری که می‌کند slave است) پس CPU می‌گوید من
تا مدت‌ها رئیس بودم الان می‌خواهم بخوابم پس می‌رود در وضعیت float می‌رود
می‌گوید دیگر از این خط‌ها را من کار نمی‌کنم. پس data bus در وضعیت float قرار می‌گیرد
در لحظه (*) همچنین مهم است چون ممکن است خانه مشخص نزده و در لحظه (**) CPU
data bus خودش را پس کرده پس خط Read فعال می‌شود و در حافظه باز می‌شود و حافظه
اطلاعات آن خانه‌ای که مشخص شده را داده روی data bus قرار می‌دهد پس data bus
همه‌ی بیت‌ها را می‌گیرد و data داخل CPU می‌رود. حالا کارمان تمام است خط
RD را غیرفعال می‌کنیم و مانند data bus ، write تا مدتی می‌شود و بعد از مدتی بعد
از آن address bus نگه داشته می‌شود.
خطی دوم شکل جزوه دیدیم که همیشه یا ما بین CPU و حافظه رد و بدل می‌شود
یعنی اوقات لازم است یا ما به دنیای خارج فرستاده شود یا از دنیای خارج گرفته
شود. اصلاً یک کامپیوتر بدون ارتباط با دنیای خارج هیچ معنی ندارد. این کار از طریق Port ها انجام
می‌گیرد. جلسه‌ی قبل بنای واحد I/O را گذاشتیم که مستقل از تمام قسمت‌ها یا Port ها از طریق
این Port ها می‌توانند بیرون و برعکس وارد شوند مانند اسپیکر که گوش آن Port
خروجی و دهان آن Port ورودی است. کامپیوتر هم به همین شکل است حالا یا از
مانیتور که از طریق Port ها می‌آیند یا از طریق data bus وارد CPU می‌شوند و سپس تبدیل می‌کنیم.
باید به Port ها هم رسیدگی شود



Scanned by CamScanner

همان طور که در قسمت اول شکل (I) می بینید در لحظه‌ی اول روی خطوط آدرس باس یک سری چیزا وجود دارد که این چیزا هر یک عددی مانند 566 روی address bus ظاهر کرده یعنی خانه‌ی 566 انتخاب شده اما خانه‌ی 566 خانه مورد نظر ما نیست. CPU قرار نیست در خانه‌ی 566 اطلاعاتی را خیره کند. در یک خانه‌ی دیگر این کار را می خواهد انجام دهد. پس در این لحظه آدرس یا نامعتبر یا invalid است. در همان لحظه (قسمت اول شکل II) روی data bus هم چیزی هست که ما آن را data می نامیم. همچنین وضعیت خاصی روی خطوط data Bus هم یک سری چیزا وجود دارد. که مثلاً شکل دهنده‌ی عدد 151 است که این عدد 151 نامعتبر یا invalid است.

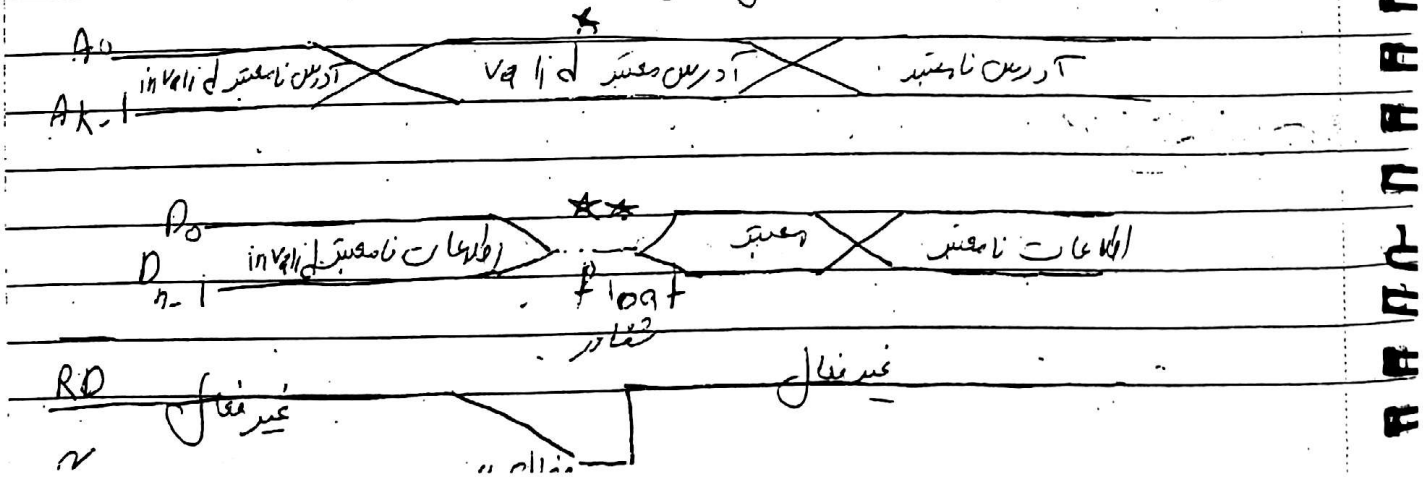
چون قرار نیست که ما 151 data را در خانه‌ی 566 ذخیره کنیم ما می خواهیم در اصل عدد 1E را در خانه‌ی 263 ذخیره کنیم. مطابق شکل اولین کاری که CPU انجام می دهد این است که (*) در لحظه‌ی اول address bus تغییر وضعیت داده و آدرس معتبر یا شماره خانه‌ی معتبر روی خطوط address bus قرار می گیرد که همان 263 است. به بیان سادگی آدرس معتبر یا valid یعنی آدرس در لحظه (**) data معتبر (valid) یعنی CPU می آید روی data bus خودش آن را روی خانه‌ی 263 راست در آن خانه ذخیره کند که عدد 1E است.

حال در این جا هم خانه‌ی مورد نظر توسط CPU مشخص شده هم چیزی که قرار است داخل آن خانه برود حالا در شکل III خط WR روی 1 است (چون همانطور که گفتیم فعال است) در یک لحظه که همه چیز مهیا است خط WR یک می شود یعنی می گوید آن چیزی که الان برایت آماده درست است بیا و آن را در خودت ذخیره کن (عدد 1E را در خانه‌ی 263 خودت ذخیره کن چون به حافظه وصل است) پس عمل WR انجام می شود. پس از آن دوباره می آییم WR را غیرفعال کنیم که چیزی دیگر وارد حافظه نشود.

* همیشه چیزهایی اطلاعاتی وارد حافظه می شود زمانی که خط WR حافظه فعال شود.

شرط غیر فعال شدن خط data write این است که نامعتبر شود یعنی CPU می تواند هر چیزی دیگری را روی data bus قرار دهد که اطلاعات به آن چیزی می رسد data نامعتبر و آشوبناک (***) می شود چون write غیرفعال شده و بعد از آن به ما هیچ ربطی ندارد که آدرس نامعتبری شود

** در این شکل timing خیلی مهم است که در هر زمان چه اتفاقی می افتد.
خود وضعیت سیگنال های CPU در حال Read:



Scanned by CamScanner

در این شکل همه حافظه‌ها به Data bus متصل شده (خطوط مربوط) و همین Data bus به Path ها متصل شده (خطوط آبی) حالا این Path ها می‌توانند Path های ورودی و یا خروجی باشند چون دیتا هایی که از این Path ها رد می‌شوند یا از طرف CPU یا حافظه‌ها می‌آیند و یا برعکس یعنی دیتا های حافظه‌ها CPU می‌توانند به یکی از Path ها ارسال شود. (حتی Path ها باید به data bus متصل شوند) ما در یک کامپیوتر تعداد زیادی Path داریم و فرض کنید می‌خواهیم اطلاعاتی را به بیرون بفرستیم و یا از بیرون بگیریم. باید ببینیم از طریق چه Path ای اطلاعات وارد یا از طریق چه Path ای اطلاعات خارج شود. حال چگونه در این کار انجام می‌شود؟ همانطور که حافظه Path ها هم مشخص دارند که اصطلاحاً به آن‌ها Path ها آدرس Path می‌گویند. طبیعی است زمانی که دیتا ها می‌خواهند بیرون خارج بروند یا برعکس باید آدرس پورت از طریق address bus مشخص شود (به شکل دقت شود که address bus علاوه بر این که به حافظه متصل است به Path ها هم وصل است)

فرض می‌کنیم روی خطوط address bus عدد ۶۵ قرار گیرد. هم خانه‌ی ۶۵ انتخاب می‌شود هم پورت ۶۵ (همین که در کل لحظه فقط یکی از آن‌ها منظر باشد یعنی یا می‌خواهیم نقل و انتقال با خانه‌ی ۶۵ حافظه داشته باشیم یا نقل و انتقال با Path انجام شود. حالا چکار کنیم؟

در سیستم‌های کامپیوتری واحد I/O به ۲ روش انتخاب می‌شود. (سوال امتحانی)

۱- روش I/O جدا (Isolated I/O)

۲- روش نگاشت حافظه (Memory mapped I/O)

پس وقتی عدد ۶۵ روی address bus از ما می‌خواهیم منظور پورت ۶۵ است یا خانه‌ی ۶۵؟ بستگی به روش پیاده‌سازی I/O دارد. به روش اول یا دوم. روش (Memory mapped I/O)

در این روش مزایای سیستم فضای آدرسی را روش نگاشت خانه‌ها می‌توانند تقسیم‌بندی کرده و تعدادی از شماره‌ها (آدرس‌ها) را برای حافظه و تعدادی از شماره‌ها را برای Path ها در نظر گرفته پس خودتان شماره‌ها را بدهید که منظور حافظه است یا Path است. به مثال زیر توجه کنید

فرض می‌کنیم address bus ما ۱۶ بیتی باشد. (A_0 تا A_{15}) با این A_0 تا A_{15} ۱۶۵ تا عدد می‌توانیم تولید کنیم ما در این ۱۶ عددی که هست فضای آدرس CPU ما ۱۶ است. فرض می‌کنیم طراحی می‌کنید عدد ۵ تا ۱۲ برای Path ها و عدد ۱۱-۰ برای حافظه باشد. حالا آدرس‌هایی از خانه‌ی ۵ تا ۱۲ روی ۹ بخواهیم از Path ها خوانده می‌شود اگر بخواهیم از خانه‌ی شماره‌ی ۱۳ بخوانیم از Path ها خوانده می‌شود به این روش روش نگاشت حافظه می‌گویند پس فضای حافظه را فدا کردیم. حال در حالت عادی این CPU می‌توانست یک حافظه‌ای به اندازه‌ی ۱۶ بیت به آن وصل شود اما الان حافظه ۱۱ خانه دارد که این نوع ایراد است

Scanned by CamScanner

	A_3	A_2	A_1	A_0	
حافظه	0	0	0	0	= 0
	0	0	0	1	= 1
	0	0	1	0	= 2
	0	0	1	1	= 3
	0	1	0	0	= 4
	0	1	0	1	= 5
	0	1	1	0	= 6
	0	1	1	1	= 7
	1	0	0	0	= 8
	1	0	0	1	= 9
	1	0	1	0	= 10
	1	0	1	1	= 11
Path	1	1	0	0	= 12
	1	1	0	1	= 13
	1	1	1	0	= 14
	1	1	1	1	= 15

روش I/O مجزا (Isolated I/O)

در این روش در CPU علاوه بر سیگنال RD و WR از یک سیگنال دیگر بنام IOR و IOW استفاده می‌کند. این دو سیگنال چیزی شبیه Path داخلی خود (دیتا، آدرس یا کنترل) هنگامی که سیگنال RD، WR فعال شود حافظه عکس العمل نشان می‌دهد و آن باز می‌شود. یعنی است data داخل آن بیرون یا از آن بیرون بیاید و هنگامی که IOW و IOR فعال شود در Path قابل بررسی شود که اطلاعات ممکن است وارد یا خارج شود بستگی به این دارد که کدام فعال شود. بین دو روش I/O Isolated می‌توانیم به تعداد خانه‌های حافظه Path داشته باشیم و دیگر فضای آدرس تلف نمی‌شود. پس تعداد خطوط از Path و نوشتن در Path مانند خود RD و WR می‌باشد. با این تفاوت که به‌جای RD، WR، IOR و IOW فعال می‌شود. کاربردی‌تر که استفاده می‌کنیم I/O Isolated است.

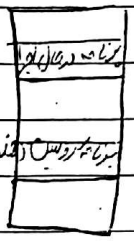
IOR و IOW هم جزء control bus هستند.

وقفه (Interrupt):

این امکان وجود دارد که در دستور حافظه چندتا بزنامه وجود داشته باشد که البته CPU فقط یکی از آن‌ها را اجرا می‌کند و بقیه در حافظه هستند

Scanned by CamScanner

وقفه چیست: اگر CPU به دلایلی برنامه‌ی درحال اجرای خودش را موقتاً رها کند و برنامه‌ی دیگری که در حافظه است را اجرا کند اصطلاحاً می‌گوییم CPU دچار وقفه شده است و بعد از اجرای برنامه‌ی دوم CPU برمی‌گردد و برنامه‌ی اول را از همان جایی که رها کرده ادامه می‌دهد.



یک برنامه‌ای که در اثر بروز وقفه اجرا می‌شود (ISR) Interrupt service Routine می‌گویند یعنی به برنامه‌ی دومی برنامه‌ی ISR می‌گویند.

X وقفه دو وقفه هم می‌تواند اتفاق بیافتد یعنی ISR در حال اجرا است دوباره خودش وقفه بخورد و یک ISR دیگر را اجرا کند

علل بروز وقفه:

۱- نرم‌افزاری ۲- سخت‌افزاری داخلی (Internal) ۳- سخت‌افزاری خارجی (External)

وقفه‌های نرم‌افزاری:

هر CPU ای می‌تواند تعدادی دستور اجرا کند. مثلاً همه‌ی CPU ها دستوری به نام دستور وقفه یا Int 21h دارند. فرض می‌کنیم در برنامه‌ی نرمی هرکدام از خط‌ها یک دستور باشند من برنامه‌نویس در داخل برنامه‌ای که می‌نویسم دستور وقفه را نیز گنجانده‌ام. پس وقتی برنامه اجرا می‌شود دستور چهارم اجرا می‌شود دستور پنجم را خودم گفتم که وقفه بخورد پس CPU اجرای آن را رها می‌کند برای اجرای ISR می‌رود و دستور برنامه‌نویس بعدی دستور پنج را اجرا می‌کند پس از آن دوباره دستور پنج را اجرا می‌کند و به همین ترتیب. با این نوع وقفه که وقفه‌ی نرم‌افزاری می‌گوییم دستور برنامه‌نویس که علت بروز وقفه شده برنامه‌نویس است که خودش داخل برنامه‌ی خودش وقفه ایجاد می‌کند. بعد CPU آن را می‌تواند این باشد:

همه‌ی میکروپروسسورهای یک سیستم عامل دارند که به واسطه‌ی سیستم عامل می‌توانیم از سخت‌افزار استفاده کنیم تا حالا به وسیله‌ی کارت شبکه‌ای از امکانات مادربورد خود استفاده می‌کردیم اما گاهی اوقات از داخل برنامه‌ی خودمان می‌خواهیم که از یک سخت‌افزار استفاده کنیم مثلاً در DVD writer بازی بنویسیم چگونه؟

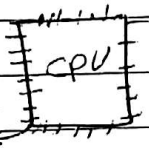
در سیستم عامل داخل خودش برای برنامه‌ی ISR دارد که هرکدام از این ISR ها برای استفاده از یکی از این سخت‌افزارها نوشته می‌شود که اصطلاحاً به آن سرویس‌های سیستم عامل گفته می‌شود. حال برای استفاده از یکی از آن سخت‌افزارها کافی است ISR را از طریق وقفه‌ی نرم‌افزاری اجرا کنیم.

Scanned by CamScanner

* * بعضی از برنامه های ISR موجود در حافظه برای سیستم عامل نیست
* بعضی از virus ها همان برنامه های ISR است

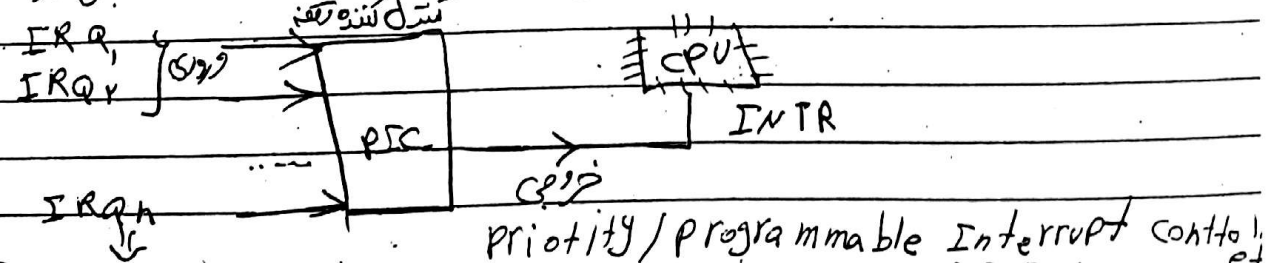
وقفه های سخت افزاری داخلی (Internal)
علت آن به وجود آمدن شرایط خاص در داخل CPU می باشد. مثلاً بالا رفتن حرارت CPU از یک حد معین یا مثلاً تغییر بیت خاص در داخل CPU

وقفه های سخت افزاری خارجی (External):
همه ی CPU ها یک سری پایه دارند که برخی مربوط به address bus و data bus و برخی مربوط به control bus است. حالا همه ی CPU ها حداقل یک یا چند پایه دارند که مربوط به وقفه ها است. اگر این پایه ها را یک بشود یعنی تغییر وضعیت پیدا کنند (از صفر به یک یا از یک به صفر) CPU وقفه می افتد که به آن وقفه ی سخت افزاری می گویند.
یعنی به وقفه ای که در اثر تغییر وضعیت این پایه ها به وجود می آید وقفه ی سخت افزاری خارجی می گویند



keyboard یا mouse یکی از پایه های CPU متصل است و هر زمانی که دکمه ای را فشار می دهیم برنامه ی ISR آن را دریافت کرده data را می گیرد (پس از فشردن دکمه یک کد دارد CPU می شود یعنی keyboard به CPU وقفه می زند و آن را مطلع می کند.)
اصالت وقفه ها آن قدر سریع است که یک لحظه جرای یکدیگر به عقب برمی گردد
یا وقفه همزمان اتفاق نمی افتد. بلکه روش time sharing است. ولی ما این گونه فکر می کنیم، تعداد پایه هایی که CPU ها برای وقفه دارند حدود است. (صد اکثر دویست) درصورتی که متناسبی بتوانند از بیرون وقفه بزنند یا دهشش برای حل این مشکل در سیستم های کامپیوتری از یک سخت افزاری بنام PIC استفاده می کنند که تعدادی پایه ی ورودی دارد که خروجی PIC به پایه ی Interrupt مربوط به CPU متصل است. هرکدام از دستگاه ها به آن IRQ می دهند یکی از آن سخت افزارها می تواند وقفه بزند حل است. مثلاً یکی به keyboard یکی به mouse و ... حالا هرکدام از این سخت افزارها که درخواست وقفه کند اول وقفه را به PIC اعلام می کند و آن از طریق پایه ی خروجی وقفه را به CPU اعلام می کند حالا چرا به آن Priority Interrupt controller می گویند؟

چون وقفه ها این به روی پایه های ورودی این می آید اولویت بندی شده اند و قابل برنامه ریزی اند



Scanned by CamScanner

چگونه تشخیص می‌دهد؟ ۱- ورودی‌ها هرکدام چه اولویتی دارند که خود اولویت‌بندی آن‌ها از ۲ جنبه است مثلاً
فرض می‌کنیم که اولویت IRQ_1 از IRQ_2 بالاتر است. حالا فرض می‌کنیم وقفه‌ای از طریق IRQ_1
می‌آید. PIC آن را به CPU اعلام می‌کند و CPU آن را اجرا می‌کند.
قبل از تمام شدن وقفه IRQ_1 وقفه‌ای از طریق IRQ_2 می‌آید حالا در این حالت دو حالت داریم:
۱- اگر گفته باشیم IRQ_2 اولویت بالاتری دارد. PIC وقفه را به CPU اعلام می‌کند که می‌گوید:
وقفه تو وقفه یعنی ISR اولی را رها می‌کند سراغ دومی می‌رود. (با
۲- اگر گفته باشیم که اولویت IRQ_1 بالاست. IRQ_2 می‌آید PIC آن را در دل
خودش نگه می‌دارد وقتی که اولویت CPU IRQ_1 را تمام کند و بعد سراغ این بیاید.
* اولویت‌ها را همواره سیستم عامل تعیین می‌کند.

Scanned by CamScanner

جلسه ی سوم

۹۵/۷/۲۴

وقفه به این شکل بود که پردازنده اول CPU برنامه در حال اجرا را رها می کرد و می رفت سراغ برنامه های دیگری که وقفه داشت اجرایی کرد سپس به ISR می آمد و بعد از اتمام شکل آن به برنامه قبل برمی گشت و آن را اجرا می کرد.

توجه وقفه ISR های متفاوت است.

Mask بیت وقفه ۳ تا هست.

۱ وقفه نرم افزاری: علت وقفه خود برنامه نویس بوده است

۲ وقفه سخت افزاری داخلی: علت وقفه به وجود آمدن شرایط خاص در CPU مثل بالا رفتن حرارت یا اضرورت یکدل

۳ وقفه سخت افزاری خارجی: تحریک یکی از پایه های مربوط به وقفه در CPU است.

هر CPU یک یا چند پایه دارد برای سخت افزار خارجی تعداد پایه ها محدود است. ما پایه ها مجبور نیستیم یک سخت افزار مخصوص به نام PIC وصل می کنیم تعدادی ورودی دارد به نام IRQ و خروجی دارد که به پایه ی CPU وصل است قابل برنامه ریزی است. از دو جنبه پیکربندی می شود.

جنبه ی اول IRQ هایی که PIC از یک جنبه ی دیگر قابل برنامه ریزی است

سخت افزار PIC از طریق Data Bus به CPU وصل می شود یعنی وقفه از طریق یکی از IRQ ها به وجود می آید. این PIC یک عددی را به CPU می فرستد از طریق Data Bus عددی که ما مقابل Programs کردیم

مثال: برنامه ریزی کردیم اگر IRQ۰ آمد عدد ۲۵۵ بفرست اگر IRQ۱ آمد عدد ۲۵۵ اگر IRQ۲ آمد عدد ۹۰۰ بفرست و ... اول کسی که PIC را config کرده که مجدداً config PIC برعهده ی سیستم عامل است البته اگر ما برنامه نویس بتوانیم اسمبلی تسلط داشته باشیم و داشتن فایل رو در برنامه ی PIC داشته باشیم می توانیم PIC را برای سایر نیاز خودمان Program کنیم.

PIC از دو جهت برنامه ریزی شده است.

۱- هم از اولویت IRQ ها

۲- هم از تعداد عددی که هنگام تحریک هر IRQ به CPU ارسال می شود.

کاربرد وقفه نرم افزاری: از طریق داخل برنامه بخواهیم از سخت افزار خاص استفاده کنیم

کاربرد سخت افزار داخلی: دست ما نیست و برای کارهای اضطراری پیش بینی شده است مثل حرارت

یکی از کاربردهای سخت افزار خارجی:

یکی از کاربردهایش پورت ماست

Scanned by CamScanner

در هر سیستم کامپیوتری تعدادی پورت داریم مخصوص پورت های ورودی و این پورت ها ورودی برای این پیش بینی شده بودند که Data یا اطلاعات بتوانند از دنیای خارج وارد CPU شوند. حالا از دنیای خارج یعنی اینترنت یا از طرف یک کامپیوتر دیگر بیایید یا از طرف یک دستگاه دیگر مثل کیبورد.

روش برقراری ارتباط با پورت های ورودی به دو نوع است

زمانی که ما با کامپیوتری کار می کنیم انتظار داریم هر لحظه از اطلاعاتی از دنیای خارج از طرف کی از پورت ها به کامپیوتر برسند.

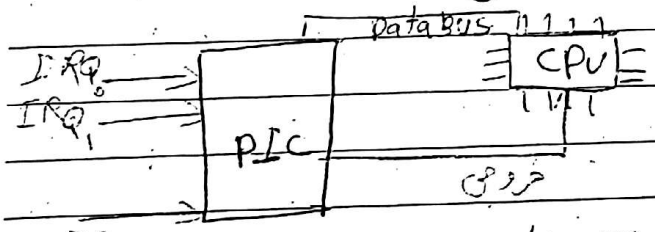
۱- روش سرکشی Poling :

در این روش CPU به طور چرخشی در فواصل زمانی منظم پورت های ورودی را چک می کند که آیا اطلاعاتی آمده است یا نه. که بعد از سپری شدن یک دوره از اول

ایراد این است که زمان بر است بنابراین زمان زیادی CPU تلف میشه مشغول چک کردن وظایف پورت ها.

۲- از این رو در سیستم ها معمولاً از این روش Poling استفاده نمی شود و برای برقراری پورت ها از روش وقفه استفاده می کنند.

به این شکل که برای هر پورت ورودی به یکی IRQ ها وصل است زمانی که از دنیای خارج اطلاعاتی به پورت رسید، این پورت از طریق IRQ مربوطه CPU وقفه می زند و این پورت مربوط میره اطلاعات رو میگیره یعنی CPU وقت آن تلف نمی شود. مثلاً می گوید هرکس با من کار داشت متوصل می زنه این روشی است که خیلی از پورت ها از آن استفاده می کنند.



روش پیاده سازی پورت
- ایزوله (جدا)
- memory mapped (داشتن)

روش برقراری ارتباط با پورت ها
- سرکشی
- وقفه ای

Scanned by CamScanner

داخل حافظه‌ی ما چیزها ISR های زیادی داریم و هر زمانی که وقفه‌ای اتفاق می‌افتد یکی از این ISR ها باید انتخاب شود و اجرا شود.

سوال: CPU چطوری انتخاب می‌کند که کدام ISR ها را اجرا کند؟

اگر ما CPU را در نظر بگیریم فرض کنید الان این حافظه با شکل (*) و ما داخل آن ۳ برنامه ISR (یعنی وقفه) داریم. حالا فرض کنید که مشاهده می‌کنید. هر ISR یک جایی از حافظه را اشغال کرده در این جا

ISRX برنامه‌اش از حافظه شروع می‌شود ۱۴۲۰ تمام می‌شود
ISRY ″ ۳۴۷۸ ″ ۵۴۱۲ ″
ISRZ ″ ۶۴۳۲ ″ ۷۶۵۴ ″

1367 3478	132
.....	1367
ISRX	1420
...	3478
.....	5412
ISRY Goto 1367	
.—	
----	6432
ISRZ	7654 (*)

هر ISR یک آدرس شروع دارد زمانی که CPU بخواهد یک ISR اجرا شود CPU باید آدرس شروع آن را داشته باشد.

سوال نوع دیگری از سوال بالا: در مثال نمونه وقتی آدرس برنامه ISR چطوری است می‌آید. همه‌جا در سیستم کامپیوتری معمولاً متناظر با شماره است. این شماره یک وقفه در واقع آدرس مستقیم یا آدرس غیرمستقیم ISR است.

منظور از آدرس مستقیم و آدرس غیرمستقیم چیست

آدرس مستقیم: مثال آدرس شروع است.

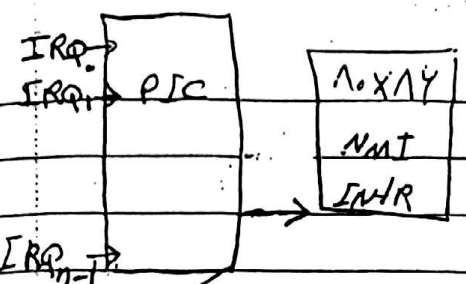
آدرس غیرمستقیم: مثلاً بردو خونه ۱۳۲ حافظه برو بیبن کجاست برمیگرد آدرس

غیرمستقیم ISRX است چون در خانه‌ی ۱۳۲ ، 1367 است که آدرس

Scanned by CamScanner

ISRX است. وقتی یک وقفه اتفاق می افتد بالای این شماره مشخص می شود.
این شماره وقفه چه‌ری تعیین می شود.
در وقفه های نرم افزاری شماره وقفه را خود برنامه توسین حاوی دستور خاصی که مثلاً (INT n) در وقفه های سخت افزاری داخلی این شماره وقفه از پیش تعیین شده است مثلاً یک CPU
مرحله خاصی اجرا می شده مثلاً داغ کند وقفه ی ۴۳۵ را اجرا می کند. یا آدرس مستقیم است
یا غیر مستقیم است که آن هم دست کم اجرا است.
وقفه سخت افزاری خارجی:
بعضی از پایه ها مستقیماً به سخت افزار داخلی ارسال می شود مثلاً یک پایه در نظر می گیریم وقتی اول
تغییر کند وقفه ایجاد می شود.
پایه
برخی دیگر از پایه های CPU وقتی وقفه از اول پایه CPU منتظر می ماند تا شماره وقفه از
طریق DataBus داده شود که شماره وقفه را از PIC می دهد.
عملکرد بعضی از سیروس های کامپیوتری:
خیلی از سیروس های کامپیوتری دارای برنامه ISR هستند و بر اساس مکانیزم وقفه استفاده
می کنند. وقایع که در ISR های سیستم عامل می نویسند.
فرض کنید که ISRX مربوط به ISR مربوط به سیستم عامل مثلاً موس باشد. اگر موس وقفه
بدهد موس وقفه می زند و این حرکت موس اجرا می شود. فرض کنید آدرس های ISR آدرس غیر
مستقیم است.
شماره وقفه نتیجه هر بار موس تکان داده شود وقفه ۱۳۲ اجرا میشه در داخل آن میگه برو
به ۱۳۶۷ و اجرا کن حالا سیستم می فهمی شده و این ویروسه می خواد بره جای
وقفه موس بنشینه، فرض کنید ISRY سال ویروس باشد چون می خواد بره جای وقفه
موس بشینه میاد آدرس خودش که ۳۴۷۸ می باشد را میبره تو خونه ۱۳۲ می گذارد
جایگزین می کند حالا کار ISRY چیست؟ هر بار اجرا شود یک فایل را پاک می شود مثلاً
موس را تشکیل داد شماره وقفه ۱۳۲ اجرا میشود وقتی اجرا ۳۴۷۸ اجرا کن یک فایل
می بره حالا ویروس ها اینطوری عمل می کنند که ما از وجودشان مطلع نشویم. اولاً موس
تکان داده شود تکان نمی خورد ISRX اجرا شده و ما می فهمیم که ویروس نزده ولی ویروس
برای اینکه ما نفهمیم میاد ISRY که میره به ۱۳۶۷ Jo Fo می زنه که می بره
او جا که موس تکان بخورد و ما متوجه ویروس نشویم.
وقفه در کامپیوترهای شخصی:
CPU های که در این کامپیوتر به کار رفته از پرداز نده ی خانواده Intel ۸۰x۸۶ استفاده می کنند.
Intel وقتی CPU جدیدی را وارد بازار کرد. این CPU جدید با مدل قبلی کاملاً سازگاری
داشت یعنی برنامه های قبلی هم کار می کرد هیچ نیازی به تغییر نرم افزار نبود.

۹۶



این CPU ها دو تا پایه برای دریافت وقفه خارجی دارند.

۱- NMI ← Non maskable Interrupt

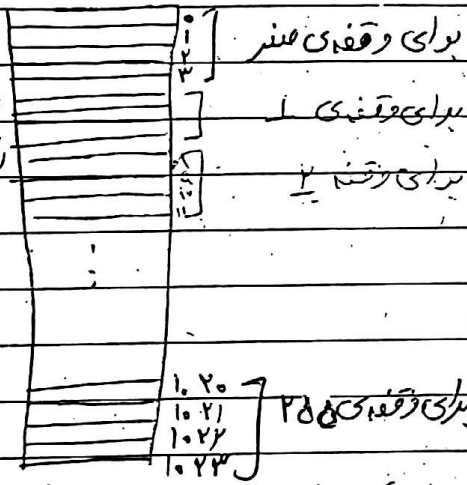
۲- INTR

پردازنده های خانواده ۸۰X۸۶ ۲۵۶ تا شماره بردار می پذیرند. هر وقفه ای که ایجاد میشود شماره اش می تواند ۰ تا ۲۵۵ باشد. اگر وقفه ای از طریق NMI به صورت خودکار وقفه ۲ است.

* اگر وقفه ای INTR میاد CPU منتظر می ماند تا از PIC شماره بیاد.

* پردازنده ۸۰X۸۶ از آدرس غیر مستقیم استفاده می نماید (به معنی از حافظه مراجعه می کرده تا آدرس ISR در آن جا بوده است)

* آدرس هر برنامه ISR ۳۲ بیتی است. در ۸۰X۸۶ که می خواهد به حافظه وقفه ای شماره ۵ تا ۴ خونه اول آدرس وقفه ای مذکور ذخیره شده



جدول بردار وقفه
Interrupt vector table

این شماره وقفه ها همیشه در ۴ ضرب میشه و به ۴ محل پشت سر هم مراجعه می کند تا آدرس ISR را پیدا کند

نقاط شروع تا ۴ تا بعد هر کدام

وقفه ۰ ← ۰X۴ = ۰
وقفه ۱ ← ۱X۴ = ۴
وقفه ۲ ← ۲X۴ = ۸
وقفه ۲۵۵ ← ۲۵۵X۴ = ۱۰۲۰

کل این فضا ۱KB است.

از ۵ تا اول ۲۵۶ وقفه مقال سیست، وقتی فعال شد دیگر روی آدرس ندارد.
یکسری از این وقفه ها را برای ما BIOS گذاشته از در جدول بردار وقفه

Scanned by CamScanner

وقفه‌های داس:

اولین وقفه‌های ESR آی‌هابرای سیستم عامل است.

قسمتی از فضای حافظه از جنس Rum است.

Rum ها می‌توانند بدون نیاز به برق محتویات خود را حفظ کنند مثل Flash memory

قسمت از فضای کامپیوتر شما از Rum است که بایاس می‌کنید اولین برنامه‌ای که کامپیوتر را روشن می‌کنیم در خیره می‌شود. قبل از ویندوز بایاس اجرا می‌شود.

در داخل بایاس مقداری برنامه ESR دارد. مقداری بایاس داخل سیستم عامل است

یک مقدار ESR داخل سیستم عامل داریم مقداری از این ESR ها داخل سیستم عامل است

این هارد دیسک ما هستند از روی دیسک خوانده می‌شود میاد توی رم قرار می‌گیرد.

آدرس هایی که سیستم عامل می‌گیرد میزارد بردار وقفه.

مقداری دیگر این ESR در بایاس است. وقتی کامپیوتر روشن می‌کنیم اول بایاس فعال می‌شود.

وقفه بایاس: اولین وقفه‌هایی هستند که ESR آی‌ها در Rum است.

آدرس آن‌ها را برنامه بایاس خودش اول را در جدول بردار وقفه قرار می‌دهد.

وقفه داس: اولین وقفه‌ها هستند که ESR روی دیسک است. وقتی ما سیستم را روشن می‌کنیم یا ضربه سیستم عاملی که میاد در حافظه قرار می‌گیرد ESR هم میاد در حافظه قرار می‌گیرد. آدرس هایی خود سیستم عامل میاد توی بردار وقفه قرار می‌دهد

مقداری دیگر ESR نرمال بایاس هست نرمال داس است برای همین برنامه نویسد است

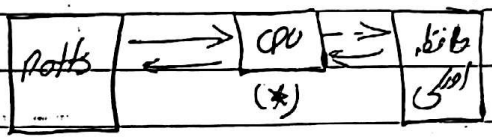
Scanned by CamScanner

جلسه پنجم ۱۸/۹۵/۹۵:

عمل DMA (Direct Memory Access):

منظور از DMA دسترسی مستقیم به حافظه است.

در بیشتر اوقات ما می‌خواهیم حجم زیادی از اطلاعات را (مثلاً از Path های ورودی) بخوانیم. برای مثال Flash Memory را به کامپیوتر متصل کردیم و می‌خواهیم یک فایل بزرگ را وارد کامپیوتر یا حافظه‌ی اصلی کنیم. در چنین مواقعی در سیستم‌های سنتی، CPU اطلاعات را بایت به بایت (یک بایت یک بایت) می‌خواند، داخل خودش می‌آورد و از داخل خودش آن‌ها را در حافظه می‌گذارد (در مقابل یک برنامه). یعنی ما در چنین مواقعی به این شکل عمل می‌کنیم، یک برنامه‌ای مانند برنامه زیر داریم: (به صورت زبان ماشین است یا نیست)



C = 0 ; آدرس محل بررسی = S

شمارنده بود A را ۱ بار
Mov [S], A
C++
S++
if C < حجم اطلاعات go to ۱

در برنامه‌ی بالا S را محل شروع در حافظه یعنی مثلاً می‌خواهیم Byte ۵۰۰۰ اطلاعات را از یک Port ورودی بخوانیم. ما در محل S به بعد در حافظه ذخیره کنیم یعنی این اطلاعات را می‌خواهیم از خانه‌ی S تا S+۹۹۹ ذخیره کنیم. چگونه این کار را انجام می‌دهیم؟

با نوشتن یک برنامه. برنامه چه کاری می‌کند؟ اول کار آدرس محل شروع در حافظه (آدرس خود S) را در C (شمارنده counter) یک متغیر است که می‌گوید از آن Byte چندم تا حالا خوانده شده، پس اول کار C = ۰ می‌باشد. بعد یک دستور سمبلیک داریم. (دستورالعمل‌ها در هر سیستم زبان‌های اسمبلی متفاوت باشد. معمولاً به یک دستور MOV یا دستور ورودی می‌گویند.) فرض می‌کنیم CPU ما برای برنامه A دارد. پس می‌گوییم: MOV کن A و شماره پورت را این دستور یعنی چه؟ Register یعنی از پورت مورد نظر یا همان پورت مورد نظر یک بایت بخوان و آن را داخل ثبات A بریز (ثبات A داخل CPU می‌باشد.)

با توجه به شکل (*) از Path یک Byte اطلاعات وارد CPU شده و داخل حافظه ذخیره می‌شود.

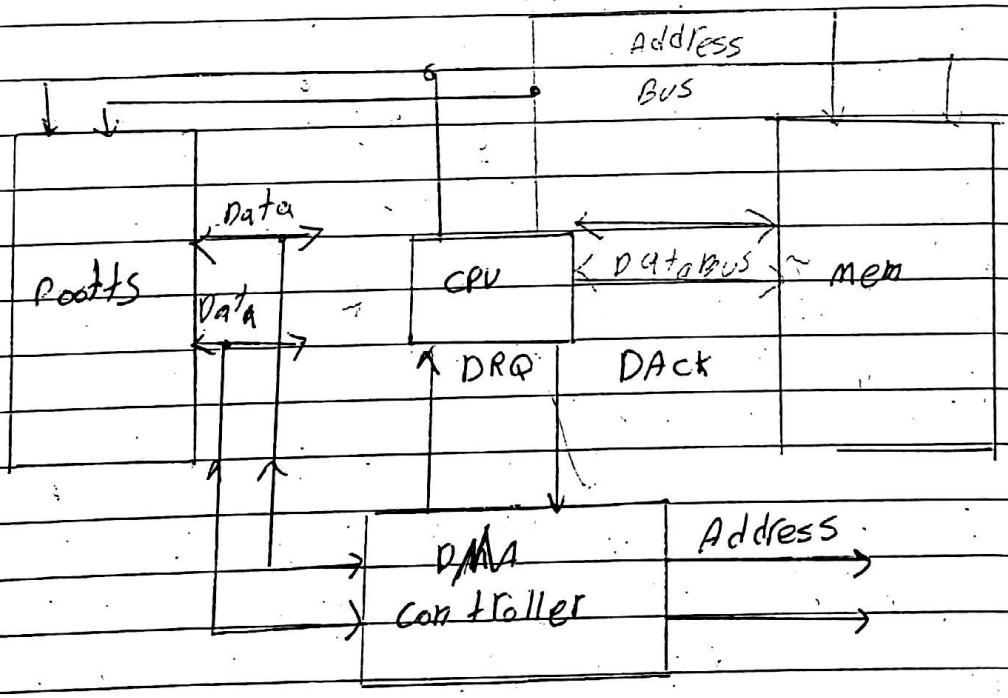
حالا می‌گوید: Move کن S و A یعنی محتوای ثبات A را در محل از حافظه که S آن را تعیین می‌کند ذخیره کن.

Scanned by CamScanner

* [ ] به معنای آدرس می باشد.

با توجه به مثال (*) اطلاعاتی که در A به 1Byte داخل شده وارد حافظه شده. در کدام محل حافظه؟ در C ؛ حالا این برای 1Byte بود. فایل ۵۰۰ بایت (به طور مثال) آن را تکرار کنیم. پس ۲ به C یکی اضافه می کنیم یعنی 1Byte به مسئله اضافه می کنیم و بعد به C هم یکی اضافه می کنیم یعنی سراغ خانه بعدی حافظه می رویم. سپس بررسی می کنیم آیا تعداد بایت هایی که تا الآن خواندیم کوچکتر از حجم اطلاعاتی است که قرار است بخوانیم. (خواننده بروده یا (در حال Loop زدن و رد شدن است) و همان کار را با اندازه حجم اطلاعات تکرار می کنیم. (این دفعه A را در C می ریزیم چون C خانه بعدی حافظه است. این کار در سیستم های سنتی اتفاق می افتد.

ایراد کل بالا؛ با توجه به مثال (*) CPU درگیر می شود. در کپی نقل و انتقال اطلاعات از Path به حافظه می شود. (در سیستم های کامپیوتری ما همیشه باید کاری کنیم که CPU درگیر عمل I/O نشود. چه کار CPU پردازش است. هر چه قدر ما CPU را کمتر درگیر کار I/O کنیم، Performance یا کارایی آن سیستم بالاتر می رود. بنابراین روش ما روش مناسبی نیست. هر چند که در سیستم های سنتی از آن استفاده می شود. برای حل این مشکل در سیستم های کامپیوتری کنترلر به نام دسترسی مستقیم به حافظه اصلی (یا DMA) تاسیس شد تا کار را راحت کند. در این روش دیگر CPU را درگیر نمی کنیم. بلکه آن را دور می زنیم. یعنی اطلاعات دیگر از داخل CPU انتقال پیدا نمی کند. از کنار CPU رد می شود. حالا به مثال زیر توجه کنید.



Scanned by CamScanner

یک سخت افزار خاص بنام DMA controller به سیستم اضافه می کنیم که مانند PIC از خانواده ی سخت افزارهای قابل برنامه ریزی است. (Programmable یعنی ما می توانیم قبل از استفاده آنها را configure کنیم تا بتوانند نیازهای ما را برآورده کنند.

(PIC از یک جنبه configure می شود و اولویت IRQ ۲ بالاتر از هر IRQ دیگری است)

DMA controller چیزهایی را نظیر آن فراهم می کند که ما در قسمت I/O ها می خواهیم از پورت های Byte مثلا ۱۰۰۰۰ اطلاعات بخوانیم و بریزیم در فلاپی دیسک به بلوک حافظه. (همچنین از اطلاعات را از یک درایو Path می خوانیم و این حجم از اطلاعات مرا است بنام فایل حافظه فرستاده شود.) حالا این DMA چه کار می کند. DMA روی همه ی CPU های که عمل DMA را support می کنند.

معمولا یک پایه ی سخت افزاری بنام DRQ (DMA request) وجود دارد، خروجی controller DMA ← عمل درخواست DMA

یک پایه ی سخت افزاری به نام DRQ ← CPU وصل می کنیم DMA controller هم با اتفاقاتی که می افتد وصل است و هنگامی که Address Bus درخواست DMA controller عمل DMA می کند. اگر CPU با عمل DMA موافق باشد یک پایه بنام DACK (DMA Acknowledge) خروجی دارد. در پاسخ به DRQ کارانجام می دهد: ۱- Data Bus و Address Bus خودش را موقت برای عمل DMA راشنود (Float) می کند یعنی ارتباط خود با Data Bus و Address Bus قطع می کند و کنترل را در اختیار DMA controller قرار می دهد. ضمن اینکه حتما DACK را فعال می کند. یعنی DMA controller می گوید کسی با درخواست DMA تو موافقت کرده و تا زمانی که DACK فعال است Data Bus و Address Bus می تواند در اختیار تو باشد. وقتی DACK آمد DMA controller اطلاعات را از Port می خواند و به داخل خودش می آورد و دستگیم به حافظه می برد در آن جا که خواست و در این مدت CPU با Data Bus و Address Bus کاری ندارد کار خودش را انجام می دهد. چرا سرعت بالا می رود؟ چون دیگر CPU درگیر نیست و یعنی خواهیم ایابت (بایت) بخوانیم و بدارد CPU کنیم و بعد از نشانی داخل حافظه بریزیم. به این عمل DMA یا Direct memory Access می گویند.

مثال: آمد CPU ای پایه ی عمل DMA را پشتیبانی نکند ما مجبوریم از روش قبلی استفاده کنیم. لذا آمد دانشگاه از هر ۲ روش می توانیم

هنگامی که یک فایل را می خواهیم از دیسک سخت بازکنیم عمل DMA صورت می گیرد. چون فایل قبلی را می گیرد و نگه می دارد چند هفته طول می کشد.

* وقتی که DRQ فعال است. CPU هم شاغل است وقتی که DMA تمام شود. DRQ را غیرفعال می کند و CPU می فهمد عمل DMA به پایان رسیده و عقلی است Data Bus را در اختیار بگیرد و کارش را انجام دهد.

Scanned by CamScanner

* عمل DMA چیست ؟ یک جمعی از اطلاعات را می خواند و در خانه های متوالی حافظه ذخیره می کند
انواع حافظه ؟

یک سیستم کامپیوتری یک CPU و حافظه ی اصلی دارد به محض این که سیستم را به برق وصل می کنیم
CPU را Reset می کنیم . CPU برای شروع به کار به محل از حافظه اشاره می کند و اولین دستوری
که می خواهد اجرا کند را از آن جا می گیرد حالا این محل از پیش تعیین شده می باشد که بستگی به نوع و مدل
CPU دارد . همگام فرض کنیم اولین بار که CPU خاص به برق وصل می شود و یا Reset می شود و سراغ
خانه ی مذا حافظه برود و اولین دستور را از آن جا بگیرد و اجرا کند . ما حافظه های اصلی
موقت هستند و تا زمانی که برق کامپیوتر وصل است اطلاعات را حفظ می کنند .
فرض می کنیم سیستم خاموش بوده و آن را روشن می کنیم . CPU همراه با خانه ی مذا حافظه
می رود و اولین دستور را می خواند و شروع به اجرا می کند . از دستوری در آن جا هست پس CPU تمرین می کند .
باید چه کار کنیم ؟ قسمی از حافظه های اصلی موقت نیستند یعنی نوع خاصی از آن ها وجود دارد که اطلاعات
درون شان به صورت دائمی است و اصطلاحا این حافظه ها را Rom می گویند .
Read only memory (Read only memory حافظه های هستند که
فقط از آن ها می توانیم بخوانیم ؛ یک بار در آن ها اطلاعات ذخیره می کنیم و بارهای بعد از آن ها استفاده می کنیم . گاهی اوقات به حافظه های Rom حافظه های WORM نیز می گویند
(Write once Read multiple) مانند CD و DVD
CD ها و DVD ها از خانواده ی Rom ها هستند و CD ها و DVD ها از خانواده های
حافظه ی جانبی هستند حافظه ی جانبی از نوع Rom هم داریم . حافظه ی اصلی از نوع Rom هم داریم
که اطلاعات درون آن ها ماندگار است .
حالا چرا در سیستم های کامپیوتری آن همه دوره ای از حافظه را CPU در هنگام انتقال برق و در اول
کار به آن ها مراجعه می کند را از نوع Rom می گیریم . و از بقیه برنامه ای در آن Rom ریخته شده
پس وقتی سیستم را روشن می کنیم CPU به آن محل مراجعه می کند تا خانه ی ۱۰۰ که این
جا از جنس Rom است و با خواندن آن چیزی از آن یاد گرفته شود . پس از وصل برق یک
برنامه در آن خانه ی ها به ما گفته است که CPU همیشه اول آن برنامه را اجرا می کند . حالا آن برنامه
بسته به نوع و مدل کامپیوتر عمل می کند .

حالا I/O چیست ؟ آیا شبیه موس و صفحه کلید و ... دیگر ها
برخی از کامپیوتر ها تک منظوره یا Special Purpose و برخی دیگر همه منظوره یا
PC
General Purpose هستند که بیشتر آن ها از این است . حالا تک منظوره فقط یک
کار خاص را انجام می دهد اما کامپیوتر همه منظوره چندین کار را می تواند با هم انجام دهد .

Scanned by CamScanner

در کامپیوترهای که منظوره خاص فقط یک برنامه قرار است اجرا شود حافظه ای کمتری در این کامپیوترها وجود دارد از جنس Rom است و نیازی به حافظه خوانای نوشتنی ندارد چون همیشه یک برنامه قرار است اجرا کند اما کامپیوتر چند منظوره به هر دو نوع حافظه نیاز دارد هم Rom و هم RAM یعنی برای شروع به Rom نیاز دارد که بگیرد حالا وقتی برنامه ی Rom اجرا شد ROM سیستم عامل ما که windows باشد را از حافظه ی جانبی می خواند و دارد حافظه ی اصلی می کند بعد از این اعمال را از طریق سیستم عامل داریم تک برنامه های دیگری را که در حافظه جانبی داریم دارد حافظه ی اصلی وصل اجرا کنیم.

(برنامه های مختلف) کامپیوترهای چند منظوره به برنامه های موجود در Rom ، Bios می گویند Bios تعدادی ESR در خودش دارد یک سری از سفت افزارهای ابزارهای این وقفه کاری کنند. (چون معمولا اول کار می کند زیر ESR آن زیر سیستم عامل است و تا زمانی که windows به کار نیفتاده کار می کند یعنی ESR آن در Bios نیست.

برنامه حافظه اصلی		CPU		

از نوع Rom

واحد I/O برای دریافت اطلاعات

کامپیوترهای از نوع General Purpose است که دو نوع حافظه اصلی دارد.
۱- خواندنی / نوشتنی ۲- Rom

خواندنی نوشتنی	
خاموش	General Purpose چند منظوره

در کامپیوترهای تک منظوره حافظه جانبی نداریم ولی در کامپیوترهای هم منظوره داریم.
* حافظه جانبی به چه دردی می خورد؟ چون در کامپیوترهای هم منظوره حافظه خوانای / نوشتنی و حافظه ای RAM داریم. آنجا به اطلاعات حافظه ای خوانای نوشتنی حرکت کند نیاز داشتیم این اطلاعات را در حافظه جانبی می بریم. اما آن قسمتی که از نوع Rom است نیازی به حافظه جانبی ندارد چون دیگر آن را پاک کنیم باز هم وجود دارد.
* برنامه ای که داخل Rom می باشد Firm ware نام دارد.

سیستم حداقلی چیست؟ دارای CPU و حافظه اصلی از نوع Rom و تعدادی Port می باشد که عمدتا برای کامپیوترهای تک منظوره استفاده می شود.

Scanned by CamScanner

حافظه‌ها را از جنبه‌های مختلفی تقسیم‌بندی می‌کنند:

۱- از نظر نحوه‌ی ارتباط با CPU و عملکردی اطلاعات ← حافظه‌ی اصلی / حافظه‌های جانبی

۲- از نظر نحوه‌ی دسترسی ← با دسترسی ترتیبی (Sequential Access Memory) SAM / با دسترسی تصادفی (Random Access Memory) RAM

۳- از نظر قابلیت نوشتن ← فقط خواندنی Read only Memory (ROM) / خواندنی/نوشتنی Read/write memory (RWM)

۴- از نظر قابلیت حمل

۵- از نظر تکنولوژی ساخت

توضیح مورد ۱

حافظه‌های اصلی، حافظه‌هایی هستند که ارتباط آن‌ها با CPU به صورت مستقیم است و محتوای اطلاعات در آن‌ها ذخیره می‌شود.

حافظه‌های جانبی، حافظه‌هایی هستند که ارتباط آن‌ها با CPU به صورت مستقیم نیست (به جز حافظه‌های مجازی) و محتوای اطلاعات در آن‌ها ذخیره می‌شود.

توضیح مورد ۲:

SAM ها حافظه‌هایی هستند که در نحوه‌ی دسترسی به محل یا اطلاعات خاصی از حافظه، دسترسی به آدرس‌ها باید از اول شروع کنیم به آن محل برسیم، یعنی نمی‌توانیم به طور مستقیم به آن محل برویم مثلاً برای رسیدن به محل ۳۰۰ باید از محل ۱ شروع کنیم تا به محل ۳۰۰ برسیم.

حافظه‌های RAM حافظه‌هایی هستند که می‌توانیم در هر لحظه سراغ هر خانه‌ای برویم (مثل RAM های فعلی)

Scanned by CamScanner

توضیح مورد ۳ :

حافظه‌های فقط خواندنی حافظه‌هایی هستند که یک بار اطلاعات برادر آن‌ها ذخیره می‌کنیم و به دفعات از آن‌ها استفاده می‌کنیم مانند CD و DVD

DVD و CD ✗ RAM هم می‌باشد. (RAM از نوع Rom که دسترسی به آن‌ها تصادفی است اما قابلیت نویسش ندارد)

توضیح مورد ۴ :

بعضی از حافظه‌ها مانند Flash memory External Hard قابلیت حمل دارند. اما برخی دیگر قطعا روی کامپیوتر نصب می‌شوند.

توضیح مورد ۵ :

بعضی از حافظه‌ها خاصیتی دارند برخی الکترونیکی برخی نوری و برخی لیزری دارند.

انواع حافظه‌های اصلی از نوع Rom و دسترسی RAM و برون نیاز قابلیت حمل :

۱ Rom

۲ PRom (Programmable Rom)

۳ E-PRom (Erasable / Programmable Rom)

۴ EEPRom = E2PRom (Electrical Erasable / Programmable Rom)

تفاوت حافظه‌های بالا ↑

۱- Rom ها با ظرفیت‌های مختلف ساخته می‌شوند و با سفارش مشتری اطلاعات در زمان ساختشان ذخیره می‌شود و غیر قابل تغییر هستند.

۲- نوع دوم که PRom می‌باشد با ظرفیت‌های مختلف ساخته می‌شوند در بازار موجود است. به شکل خام آن‌ها را خریداری می‌کنیم و برای ذخیره‌ی اطلاعات در داخل PRom ها نیاز به دستگاهی به نام Programmer داریم که به کامپیوتر هم وصل می‌شود هم اطلاعاتی که پخش داریم را یک بار می‌توانیم داخل آن ذخیره کنیم.

۳- EPRom ها هستند PRom ها هستند و در بازار به شکل خام با ظرفیت‌های مختلف وجود دارند. با دستگاه Programmer می‌توانیم داخل آن اطلاعات ذخیره کنیم اما EPRom های EPRom های حافظه‌ای هستند که می‌توانیم حافظه را با ما دقیقه در عدن نور ماورای بنفش قرار دهیم تا اطلاعاتش پاک شود.

دستگاهی به نام Erase داریم که در یک لایه ماورای بنفش است یک Timer دارد که آن را روی ۱۵ min تنظیم می‌کنیم. یک کشو دارد که Rom را داخل آن قرار می‌دهیم و در آن را می‌بندیم و دستگاه برای آن که پس از Programmer کردن نور محیط روی آن‌ها اثر نکند حافظه‌ای نیمه‌شفاف را جلوی آن‌ها می‌گذارند.

Scanned by CamScanner

EEPROM یا EEPROM یا حافظه فقط خواندنی برنامه‌پذیر با پاک شدن الکتریکی، نوعی حافظه کامپیوتر است که می‌تواند توسط programmer برنامه‌ریزی شود و داده‌ها را به صورت الکتریکی پاک کند. برخلاف ROM، می‌توان آن را بارها پاک و دوباره برنامه‌ریزی کرد. برخلاف Flash memory، EEPROM ها معمولاً فقط یک بایت در هر زمان را پاک می‌کنند و در نتیجه برای برنامه‌های کوچک که نیاز به به‌روزرسانی مکرر دارند، انعطاف‌پذیرتر است. EEPROM ها معمولاً در مقادیر کمتر از 64 کیلوبایت یافت می‌شوند. اغلب اوقات، آن‌ها برای ذخیره‌سازی تنظیمات پیکربندی سخت‌افزار در دستگاه‌هایی مانند رایانه‌های شخصی، تلفن‌های همراه و تجهیزات الکترونیکی استفاده می‌شوند. به عنوان مثال، یک EEPROM ممکن است برای ذخیره تنظیمات BIOS یک رایانه شخصی استفاده شود، زیرا این تنظیمات نیاز به به‌روزرسانی گاه‌به‌گاه دارند. EEPROM ها همچنین در سیستم‌های تعبیه‌شده مانند سیستم‌های خودرو و دستگاه‌های پزشکی استفاده می‌شوند، جایی که ذخیره‌سازی داده‌های قابل اعتماد و غیر فرار بسیار مهم است. به طور کلی، EEPROM ها انعطاف‌پذیرتر و سازگارتر با محیط‌زیست از سایر انواع حافظه فقط خواندنی هستند، اما همچنین گران‌تر و کندتر هستند.

Scanned by CamScanner

جلسه ششم ۹۵/۸/۸

انواع حافظه های اصلی از نوع Rom و با دسترسی RAM :

Rom

PRom (Programmable Rom)

EPRom = (Erasable/programmable Rom)

EEPRom = EzPRom (Electrical Erasable/programmable Rom)

مورد استفاده های این حافظه های اصلی که از نوع Rom و با دسترسی RAM هم در سیستم ها بر بستن بر نیز پرداز نده است که در واقع که ابتدای کار که این ها را روشن می کنیم چون حافظه ی اصلی مال آن نوع حافظه ای خوندنی نوشتنی با شد هیچ چیزی توش نیست. برای اینکه همون اول کار بر نامه ی بوت را اجرا کند و سیستم عامل را حافظه منتقل نشه معمولاً اول حافظه ای که همون اول کار بهش مراجعه می کنند CPU بهش مراجعه می کند از نوع RAM می گیرند. RAM ها هم خاصیت ماندگاری اطلاعات دارند چون اطلاعات پاک نشده و از قبل موجود بوده پس می تونه اون برنامه ی اولیه را اجرا کند و مشکلی نخواهیم داشت. البته اگر سیستم تک منظوره باشه معمولاً همان RAM را داریم ولی اگر سیستم ما منظوره باشه علاوه بر RAM حافظه ی خوندنی نوشتنی هم خواهیم داشت.

انواع حافظه های اصلی از نوع RWM و با دسترسی RAM :

به دو دسته تقسیم می شوند.

۱- حافظه های ایستا یا استاتیک (static)

۲- حافظه های پویا یا (Dynamic)

۱- حافظه ی ایستا یا استاتیک این حافظه هایی هستند که از عناصر فلیپ فلاپ ساخته شده اند. این ها حافظه هایی هستند که سرعتشان بالا است. اما دو تا اشکال دارند:

۱- ابعاد از نظر فیزیکی بزرگ است و برای سیستم های امروزی محدودیت ایجاد می کند.

۲- گران قیمت هستند.

۲- حافظه های پویا: این حافظه ها با عناصر خازنی ساخته شده اند که کوچک و با ظرفیت کم. به نحوی که شارژ و یا دشارژ بودن هر خازن می تواند نشان دهنده ی وجود صفر یا یک باشد. برای ذخیره یک بیت از خازن استفاده می کنیم مثلاً اگر خازن شارژ بود می گوییم 1 ذخیره شده اگر خازن دشارژ بود می گوییم بیت صفر ذخیره شده

مزایای حافظه های پویا (Dynamic)

۱- از نظر ابعاد فیزیکی کوچک هستند

۲- ارزان هستند

Scanned by CamScanner

معایب:

۱- سرعت کمتر است نسبت به استاتیک

۲- نیاز به بازنویسی یا عمل Refresh دارند.

چون بر اساس خازن ساخته شده اند خازن به صورت فایل خوانده می شوند

ما نیاز است مرتب اطلاعات آن را بازنویسی کنیم تا اطلاعات از بین نرود.

امروزه در کامپیوترهایی که استفاده می کنیم اولین حافظه ای که می بینیم که اصلاحاً RAM داریم

دنیای تجارت به حافظه ای اسمی کامپیوتر که از نوع RAM باشد RWM هم می گویند

و دسترسی تصادفی هم باشد از نوع چیپ هم باشد RAM می گویند و RAM کامپیوتر را دینامیک می سازند.

- حافظه ها چند نوع و اقسام دارد.

حافظه های نهان یا (cach)

کاملاً روزبه روز سرعت CPU در حال افزایش است و نرخ رشد حافظه اصلی آنقدر نبوده است.

سرعت حافظه اصلی جواب گو به سرعت حافظه CPU نیست.

هر کامپیوتر CPU دائم در حال تبادل صفر و یک است با حافظه اصلی. یا دستوری که یا اطلاعات می گیرد

یا نتیجه ای را ذخیره می کند. چون سرعت حافظه اصلی با CPU هم خوانی ندارد خیلی وقت ها CPU

معطل می شود و این باعث می شود سرعت کلیه کامپیوتر پایین بیاید. برای رفع این نقص پیشنهاد داده شده

یکی از این سه تا:

۱- استفاده از حافظه های نهان یا cach است.

حافظه های نهان یا cach: حافظه ای است بین CPU و حافظه اصلی قرار می گیرد و از نوع استاتیک است.

سرعتش بالا است.

- معمولاً این حافظه های نهان و cach یک cach controller دارد

cach controller با حافظه های نهان در ارتباط است

کار cach controller:

می آید بر اساس الگوریتم هایی پیش بینی می کند در آینده به چه محل هایی CPU مراجعه می کند

cach قبل از مراجعه CPU می رود اول قسمت حافظه و اول اطلاعات را می خواند و میریزد

در cach.

همیشه CPU می خواد برود از حافظه اصلی اطلاعات را بردارد اول CPU به حافظه نهان ها مراجعه می کند

به احتمال زیاد اطلاعات موجود در cach است. اگر نبود CPU به حافظه اصلی مراجعه می کند و از

آن اطلاعاتی دارد.

سرعت CPU ما آنقدر بالا رفته سرعت cach هم پاسخ گو نیست و آمدند داخل CPU

هم cach درست کردند که به آن L1 cache داخلی می گویند و به External cache یا L2 cache

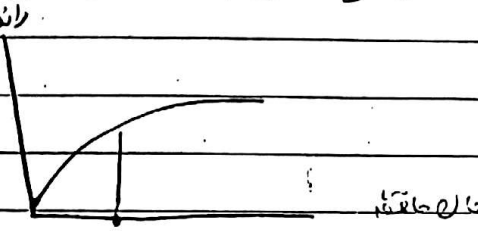
حافظه نهان بیرونی می گویند

Scanned by CamScanner

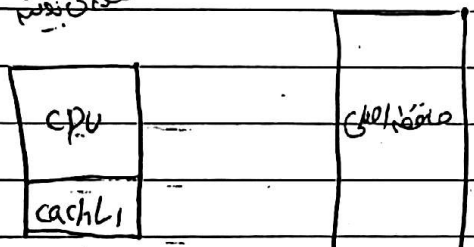
هسته cpu اول کار را برای آن آورد چون داخل خودش است. آن نیز در ram و حافظه‌ی اصلی داده‌ها باعث می‌شود سرعت سافزایش پیدا کند. سرعت آن قیمت بالاست.

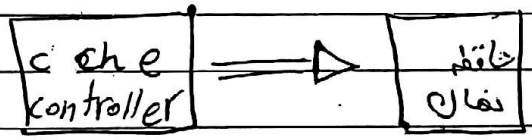
- ایزاهه نزدیک‌تر cach به معنای بهتر سرعت نیست.

شما اگر تایک جایی برای cach زیادی سرعت بالا می‌رود ولی از یک جایی به بعد شما می‌بینید زیاد کردن cach تأثیری در راندمان شما ندارد.



سیستم‌ها در زمان‌ها متفاوت است.





External cach L2

مبدل‌های آنالوگ به دیجیتال: (Analog to digital convertor) (A/D)

مبدل‌های دیجیتال به آنالوگ: (digital to Analog convertor) (D/A)

قبلی از کمیت‌ها در طبیعت وجود دارد می‌خواهیم به صورت (دیجیتال) تبدیل کنیم و در حافظه‌ی اصلی ذخیره کنیم و روش پردازش را انجام بدهیم.

این کمیت‌ها طبیعتاً آنالوگ هستند یعنی سیگنال‌هایی هستند که بستگی به زمان تغییراتشان پیوسته است. یعنی در هر لحظه از زمان این کمیت یک مقدار دارد.

وقتی ما فشار، دما، ... آنالوگ هستند جهت پردازش این کمیت‌ها در کامپیوتر باید دیجیتال شوند.

برای تبدیل آنالوگ به دیجیتال این کارها را می‌کنیم:

۱- از حسگر استفاده می‌کنیم.

حسگر یک دستگاهی است که کمیت مورد نظر ما را به یک سیگنال الکتریکی متغیر با زمان تبدیل می‌کند.

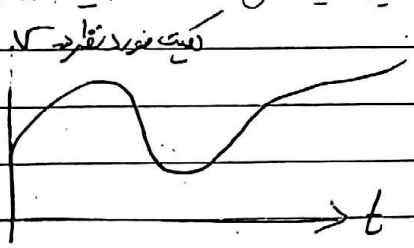
فشار هوا می‌خواهیم را دیجیتال کنیم، مولد سنگال حسگر شما میکروفون است. میکروفون امواج صوتی شما را تبدیل به سیگنال الکتریکی متغیر با زمان تبدیل می‌کند.

Scanned by CamScanner

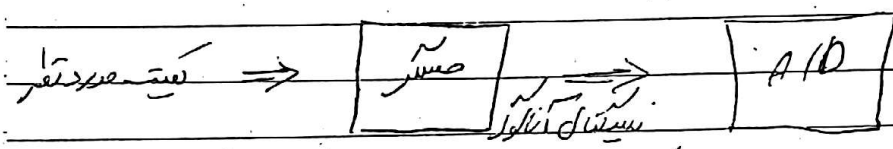
حسگرهای کلاً ماژول این است که اول کمیت معینی از طبیعت وجود دارد را تبدیل می کند به یک سیگنال الکتریکی متغیر با زمان

مثلاً حسگر نور هر چقدر نور محیط بیشتر باشد ولتاژ بیشتری می دهد هر چه نور کمتر باشد ولتاژ کمتری می دهد.

بعد از این که کمیت مورد نظر توسط حسگر تبدیل به آنالوگ شد حالا این سیگنال آنالوگ باید وارد یک مدار A/D بشود تا به شکل دیجیتال از صفر و یک می دهد.



A/D می تواند ۸ بیتی ۱۶ بیتی یا ۳۲ بیتی باشد



در اینجا فرض ما بر این است که ورودی ما به صورت سیگنال آنالوگ وارد می شود. یک از طریق این صورت سنسورهای مختلف می شود.

A/D چگونه کار می کند:

A/D در ۳ مرحله سیگنال آنالوگ را به دیجیتال تبدیل می کند.

نحوه تبدیل سیگنال آنالوگ به دیجیتال در مبدل های A/D:

جهت این امر در ۳ مرحله سیر حرکت پیش بینی شده است.

۱- نمونه برداری - در این مرحله مقدار سیگنال از ورودی (حسگر) خوانده می شود. (sampling)

۲- تقریب - زمانی مقدار خوانده شده به یک حداقل یا بین تقریب زده می شود (Approximation)

۳- مقدار تقریب زده شده در مرحله قبل با یک مقدار دیجیتال n بیتی (یک عدد n بیتی) متناظر می شود (Quantization)

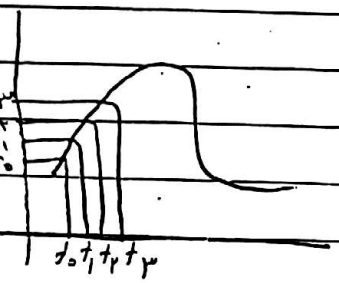
۲ مرحله ای فوق به طور منظم و با یک سرعت ثابت تکرار می گردد.

حال نحوه عملکرد تقریب زننده به یک مقدار دیجیتال متناظر می کند

همیشه اولین داستانی که داریم باید بدانیم چند bit است مثلاً n. اگر ۸ بیتی باشد در مرحله تقریب زده شده را تبدیل به یک عدد ۸ بیتی می کند و... و هر چه قدر عدد n بزرگتر باشد دقت A/D ما بیشتر و A/D ما کارآمدتر خواهد بود.

Scanned by CamScanner

همینطور تا سیگنالی که می خواهیم دیجیتال کنیم را بدست آوریم معلوم باشد. مثلاً بین ۰ تا ۵ ولت است
حرف می شنوید را می گوییم آنرا ابتدا نوانتها ندارد نمی دانیم صداقل و حداکثر آن چقدر است - باید بتوان
به دیجیتال تبدیل کرد.
سنسورها معمولاً دامنه دارند. و پیوسته اند. مساحتی داریم.
حرفی که می خواهیم به دیجیتال تبدیل کنیم



نمونه ای که برمی داریم باید به یکی از این حالت ها بریابد $V_0 = 1,016$
کمترین مقدار سیگنال ooooooo ⟸ $V_1 = 1,8\,V$
oooooo1 $V_2 = 2,26$
ooooo10 $V_3 = 3,1$

1111110
بیشترین مقدار سیگنال 1111111
۰٫۸ - ۰٫۶ - ۰٫۴ - ۰٫۲ - ۰ ← ۵ کمترین فاصله ها
$\frac{5}{256} = 0,02$
5 ... ۱٫۴ - ۱٫۲ - ۱ ...

$V_0 = 1,016 \Rightarrow 1,02$
$V_x = 0,018 \Rightarrow \%0,2 \Rightarrow$ oooooo01 از بین تا ۲۵۶ باید کدام را انتخاب کرد
و یا ۲۵۶ دوم منظور کرد.
A/D می آید دایره های زردی متعلق از سیگنال ما نمونه برداری می کند. در هر لحظه این سیگنال یک
مقداری خواهد داشت
حرفی که برای ما حائز اهمیت است برای هر مقدار اول ۳ تا عمل اجرا می شود.
حرفی که سیگنال بیشتر حداقل و کمتر حداکثر ۵ ولت است. و می خواهیم A/D ۸ بیتی باشد.
دقت اول دیجیتال ۵
۱ - هر یک A/D هر چه n بزرگ باشد دقت بیشتر
۲ - سرعت نمونه برداری
شبیه تو لحظه t_0 و t_1 برط سیستم سر این دو تا ضرب نداریم تعداد سیگنال حرفی داشتیم
این دو تا دامنه نمونه از بین رفته
Amdal :

Scanned by CamScanner

جلسه‌ی هفتم: (۹۵/۸/۱۳)

نحوه‌ی کار A/D در جلسه‌ی گذشته شرح داده شد. مثلاً برای دمای مقادیر دما را در اختلاف مختلف باید سری عددها را مقایسه کنند.

آیا ما می‌خواهیم تغییرات دما را ثبت کنیم. آیا دقیقه به دقیقه یا ثانیه به ثانیه این طور را انجام می‌دهیم؟ خیر معمولاً بسته به آهستگی تغییرات در فواصل زمانی منظم از مقدار دما نمونه می‌گیرند. مثلاً در مورد دما می‌گوییم هر ۱ ساعت ۱ نمونه ثانیه تغییر نمی‌کند. مثلاً فصل است یک ساعت، یک بار تغییر کند. پس مقدار دما را در هر یک ساعت ثبت می‌کنیم. حال اگر بخواهیم دما را ثانیه به ثانیه ثبت کنیم با حجم زیادی از اطلاعات مواجه می‌شویم که نگه داری و پردازش آن ها ممکن است خیلی سخت شود. پس در فواصل زمانی معین این کار را انجام می‌دهیم. A/D هم نیز همین کار را انجام می‌دهد. یعنی در فواصل زمانی منظم از مقدار سیگنال نمونه (sample) می‌گیرد.

حالا این فواصل زمانی منظم تغییر پذیر است. یعنی می‌توانیم بگوییم: ۱ ثانیه یک نمونه، ۱ دقیقه یک نمونه (البته بستگی دارد تغییرات با چه سرعتی صورت می‌گیرد. مثلاً فواصل زمانی منظم می‌توانیم نمونه بگیریم پس از اینکه A/D نمونه‌ها را گرفت، ۲ کار دیگر هم باید انجام دهد. برای هر نمونه اولین کار این است که مقدار نمونه به یک حدهایی پایین تر یا بالاتر تبدیل شود. چرا؟

چون خروجی یک سنسور یک عدد حقیقی است بین یک min و max.

مثلاً خروجی یک سنسور ۵ ولت است. پس خروجی این سنسور یک سیگنال الکتریکی است که یک min و max ای دارد. پس این sample هایی که در هر لحظه می‌گیریم عددی است بین min و max. پس min و max نهایت عدد وجود دارد یعنی حالت‌هایی تواند باشد که هر یک از این حالت‌ها مقدار سیگنال در آن لحظه‌ی خاص است که این مقدار سیگنال بین min و max است.

مثلاً اگر فرض کنیم خروجی سنسور بین ۵ ولت می‌باشد. نمونه‌هایی که می‌گیریم ممکن است ۲/۳ یا ۱/۴ یا ۱/۷ ولت و ... باشد. (بی نهایت حالت می‌تواند داشته باشد)

اما از طرفی چیزهایی که A/D به ازای هر نمونه تحلیل می‌کند یک عدد باینری است که این باینری می‌تواند ۸ یا ۱۶ یا ۳۲ بیت باشد یعنی حالت‌هایی که A/D می‌تواند به ازای هر نمونه داشته باشد محدود است. در صورتی که حالت‌های ورودی نامحدود است. مانند مثال جلسه‌ی پیش

* تقریب زدن باعث یک اختلال می‌شود چون مقدار واقعی با در مثال جلسه‌ی قبل ۸/۳ بود که دستگاه ما آن را ۸ گرفت. یعنی همه‌ی A/D ها خطا دارند.

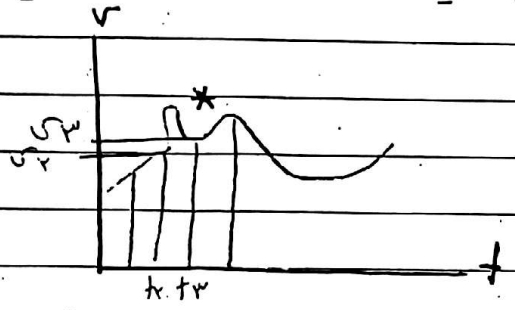
* اگر در مثال جلسه‌ی قبل به جای A/D ۸ بیتی از A/D ۱۶ بیتی استفاده می‌کردیم باید تقسیم بر 2^{16} می‌شد و فاصله‌ی step ها کمتر شد. در نتیجه خطای تقریب زدن هم کمتر می‌شود. ما به عدد A اصلاحیه دقت A/D می‌گوییم.

یعنی هرچقدر A/D بیشتر باشد، دقت A/D نیز بیشتر است. پس فیزیکی A/D ۱۶ بیتی از A/D ۸ بیتی دقت بیشتری دارد. در عمل معمولاً از ۱۲ بیتی می‌توانیم حجم اطلاعات بیشتر می‌شود. یعنی یک A/D ۱۶ بیتی ۲ برابر یک A/D ۸ بیتی دقت دارد.

Scanned by CamScanner

پس فعلی می خواهیم آن را ذخیره کنیم، حجم اطلاعات ۲ برابر می شود. یک A/D ۳۲ بیتی، ۲ برابر یک A/D ۱۶ بیتی به ما وعده می دهد. البته در شرایط مساوی یعنی سرعت نمونه برداری یکسان باشد. پس از طرفی می خواهیم حافظه تقریب بزنی را کاهش دهیم و از طرف دیگر باید حجم بیشتری از اطلاعات را در نظر بگیریم.

* در A/D بحث دیگری نیز داریم به نام سرعت نمونه برداری یعنی تعداد نمونه ای که در هر ثانیه می گیریم.



با توجه به شکل بالا، در لحظه t_2 نمونه گرفتیم و مقدارش برابر V2 شده و در لحظه t_3 هم نمونه گرفتیم و مقدارش برابر V3 شد. حال می بینیم بین لحظه t_2 و t_3 یک جهش یا خرش شدید داریم (تغییر وضعیت ناگهانی) و این نمونه (*) را از دست داریم چون مقدار سیگنال در لحظه t_2 و t_3 را می دانیم اما از مقدار سیگنال بین لحظه t_2 و t_3 هیچ اطلاعی نداریم. همین باعث بروز مشکل می شود. حال در صورتی این نمونه را از دست نمی دادیم؟ در صورتی که بین t_2 و t_3 یک نمونه دیگری داشتیم یعنی سرعت نمونه برداری بیشتر بود. فرض می کنیم ۲ برابر این سرعت نمونه برداری می کردیم به احتمال زیاد (یک) (*) از دست نمی رفت. (تعداد نمونه دوتا شد) حالا چرا مهم است؟ چون عدد (*) بخشی از اطلاعات بود که از دست دادیم. حالا می آییم بحث: هم سرعتی باید نمونه برداری کنیم تا دچار این مشکل نباشیم؟ یعنی بتوانیم ادعا کنیم که تقریبا چیزی را از دست ندادیم. هیچ وقت نمی توانیم ادعا کنیم همه چیز را نمونه گرفتیم چون بالا فرجه بین t_2 و t_3 بی نهایت کوچک داریم و ما نمی توانیم بی نهایت نمونه بگیریم ولی می آییم با چه سرعتی نمونه برداری کنیم که آنقدر خوب شود نمونه هایی که گرفتیم را بازسازی کنیم با حداقل خطا. حداقل خطا در حد نزدیک تر باشد.

دانشمندی به نام Nyquist جواب این سوال را داد. این دانشمند از نظر ریاضی آن را اثبات کرد. او گفت برای این که فرکانس تبدیل آنالوگ به دیجیتال سرعت نمونه برداری چقدر باشد باید حداقل سرعت نمونه برداری ما ۲ برابر Maximum فرکانس موجود در آن سیگنال باشد.

فرکانس موجود صدای انسان Maximum فرکانس موجود در صدای انسان ۴KB است. یعنی گوش انسان بیشتر از ۴KB نمی شنود. حالا اگر بخواهیم صدای یک نفر را به صورت دیجیتال و با کیفیت مطلوب ضبط کنیم به شرطی که هنگام پخش صدا با صدای اصلی تفریبا یکسان باشد (به دلیل تقریب زنی، هیچ گاه صدا مثل خودش نمی شود)

ما باید با سرعت ۸KHz نمونه برداری کنیم یعنی ۸۰۰۰ نمونه در هر ثانیه را باید داشته باشیم تا صدا با کیفیت مطلوب ذخیره شود. پس A/D در هر ثانیه باید ۸۰۰۰ تا نمونه بگیرد و همه اعداد را در یک فایل ذخیره می کند. حال برای Music هم همین طور است.

Scanned by CamScanner

مثلاً maximum فرکانس موجود ۲۰ kHz است سرعت نمونه برداری باید ۴۰ kHz باشد یعنی ۴۰۰۰۰ نمونه در هر ثانیه باید داشته باشد تا music با کیفیت مطلوب ذخیره شود هرچه نرخ نمونه برداری کمتر باشد کیفیت صدا کمتر است و معمولاً music ها را با نرخ نمونه برداری ۴۴۱۰۰ ذخیره می‌کنند.

* حجم داده‌ها تأثیر زیادی بر عامل بالایی دارد: ۱. دقت A/D یعنی تعداد بیت‌های هر نمونه بیشتر باشد خطای تقریب زنی کمتر می‌شود و در نتیجه حجم داده‌ها افزایش پیدا می‌کند. ۲. اگر سرعت نمونه برداری را بالا ببریم مثلاً در ۱ ثانیه به جای ۱ نمونه ۲ نمونه بگیریم حجم اطلاعات بیشتر می‌شود.

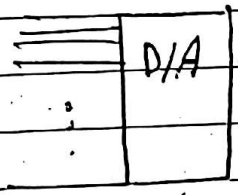
پس به دلیل ۲ مورد بالا حجم اطلاعات فایل‌های صوتی/تصویری زیاد است.

دوربین‌های قدیمی حافظه‌ها شان فیلم‌های آنالوگ بودند که فضای زیادی را اشغال می‌کردند. دوربین‌های جدید حافظه‌ها را با تکنیک‌های فشرده می‌کنند که mp3 یکی از روش‌های فشرده‌سازی است پس A/D mp3 نمی‌کند، بلکه می‌تواند داده را فشرده می‌کند و نمونه‌ها را کم می‌کند.

پس اگر سرعت نمونه برداری به اندازه maximum فرکانس موجود در آن سیگنال باشد، مشکل موج بازسازی شده به مشکل موج اصلی نزدیک‌تر است.

* گاهی اوقات ما کیفیت آنالوگ را با تبدیل به دیجیتال کردن از دست می‌دهیم و دوباره آنالوگ می‌کنیم و دوباره دیجیتال می‌کنیم و ... اگر بخواهیم این اطلاعات دیجیتال شده را دوباره به حالت اول برگردانیم مثلاً مدار A/D تبدیل به دیجیتال کرده و در کامپیوتر ذخیره می‌کنیم، حالا می‌خواهیم مدار را خنثی می‌کنیم اما مشکل برای ما این است که اطلاعاتی که ما را تبدیل به دیجیتال کرده در کامپیوتر ذخیره می‌کنیم اما اطلاعات ذخیره شده را دیگر نمی‌توانیم تبدیل به دما کنیم و نمی‌خواهیم آن دما را افزایش کنیم یعنی بعضی جاها ما اطلاعات دیجیتال را می‌خواهیم تبدیل به آنالوگ کنیم ولی بعضی جاها نمی‌خواهیم.

اگر خواستیم اطلاعات دیجیتال را به آنالوگ تبدیل کنیم از وسیله‌ای به نام digital to analogue converter (D/A) استفاده می‌کنیم.

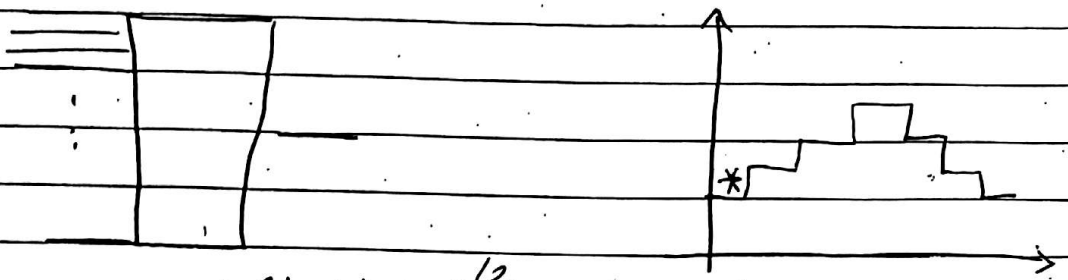


* D/A چه کاری می‌کند؟ وقتی عدد n بیتی را به D/A بدهیم و ولتاژ متناظر با آن عدد را برای ما تعلیق می‌کند. مثلاً در مثال خیلی قبل دیدیم که اگر ولتاژ ما ۱۸ ولت بود (۱۸ = ۷x ۲ + ۴) آن را تقسیم می‌زدیم و می‌شد ۲ باقی ۴ و ۲ باقی ۱ و ۱ باقی ۰ که معکوس می‌شد ۱۰۰۱۰ حالا D/A برعکس عمل می‌کند، یعنی اگر به D/A عدد ۱۰۰۱۰ را بدهیم، در خروجی ۱۸ ولت را می‌دهد. (۱۸ = ۱۶ + ۲)

پس D/A n بیت ورودی و ۱ خروجی دارد که همان ولتاژ است، یعنی نمونه n بیتی که به D/A اعمال می‌کنیم، در خروجی آن ولتاژ تقریب زده شده (نه اولیه) را به ما برمی‌گرداند.

حالا اگر بخواهیم شکل موج اولیه را بازسازی کنیم. نمونه‌های ذخیره شده در حافظه‌ی کامپیوتر را با همان سرعتی که نمونه‌برداری کرده بودیم با همان سرعت به D/A اعمال می‌کنیم مثلاً نمونه‌ی اول را اعمال می‌کنیم و ولتاژ (*) را به ما می‌دهد و همین ترتیب.

Scanned by CamScanner



شکل هم می‌که درست می‌شود پله‌ای است ولی از نظر روند تغییرات، با شکل موج اصلی فرق دارد.
حالا چرا پله‌ای است؟ مثلاً تقسیم در لحظه‌ی ۰ ← ۰/۱ ولت، در لحظه‌ی ۱ ← ۰/۲ ولت و ما
بین لحظه‌ی ۰ و لحظه‌ی ۱ از مقدار سیگنال بی‌خبریم چون از آن نمونه‌گیری نشده. پس D/A خروجی می‌کند
که مقدار سیگنال بین این ۲ لحظه ثابت بوده و تغییرش نکرده به همین دلیل شکل موج بازسازی شده
به صورت پله‌ای است. حال اگر سرعت نمونه‌برداری مناسب باشد، یعنی طبق قانون Amdal و رعایت
کرده باشیم، این شکل موج بازسازی شده تقریباً همان شکل موج اصلی نزدیک‌تر
است اما هیچ‌گاه دقیقاً مانند شکل موج اصلی نمی‌شود. چون مقادیر ثابتی که دیجیتال نشان داده
شده (خطوط صاف) تقریب‌ها هستند. مثلاً آن ۰/۲ نبود بلکه ۰/۲۰۰۰۰۰۰۰۱ بود که هیچ‌گاه
بازسازی نخواهد شد. پس کمی به این علت است که دقت A/D پایین است دیگری هم به این دلیل که ما
از مقادیر سیگنال ما بین ۲ لحظه اطلاعی نداریم. حالا این شکل موج بازسازی شده را به یک تقویت
کننده می‌دهیم و این تقویت کننده هم به یک بلندگو می‌دهیم و دوباره صدا برای ما پخش می‌شود.
حالا هرچقدر دقت A/D بالا باشد یعنی n آن بزرگ باشد و مقادیر باشد باید عین بیشتری نمونه‌برداری
کنیم و قیمت آن هم گران‌تر است.

* Maximum سرعت نمونه‌برداری بستگی به قسمت A/D دارد. یعنی مثلاً A/D بیشتر از 48000 تا نمی‌تواند.

فشرده‌سازی و دیتا

فشرده‌سازی روش‌های مختلفی دارد که به یکی از آن‌ها اشاره می‌کنیم
فرض کنید عدد دهدهی اول برابر با ۲۳۵ باشد، عدد دهدهی دوم برابر با ۲۳۸ باشد، نمونه‌ی سوم
برابر با ۲۴۰ باشد. آنچه ما می‌خواهیم هرکدام از این نمونه‌ها را به صورت فشرده کنیم (هر نمونه‌ها یک عدد
۸ بیتی است)

به صورت Binary نوشتیم:

۲۳۵	۲۳۸	۲۴۰
۱۱۱۰۱۰۱۱	۱۱۱۰۱۱۱۰	۱۱۱۱۰۰۰۰

۲۴ بیت حافظه اشغال کرد.

یک روش نمونه‌سازی می‌گوید: اولین عدد چند است؟ ۲۳۵
عدد بعدی چند است؟ ۲۳۸ ← +۳
عدد بعدی ۲۴۰ است که ۲۴۰ = ۲ +

۲۳۵	۲۳۸	۲۴۰
۱۱۱۰۱۰۱۱	۱۱۱۰۱۱۱۰	۱۱۱۱۰۰۰۰

۲۳۵ + ۳ + ۲

Scanned by CamScanner

حالا چه نرخ زده می‌شد؟ عدد اول را ذخیره می‌کنیم یعنی ← ۱۱۱۱۱۱۱

بعد ۳ تا اضافه می‌کنیم یعنی ← ۱۱۱۱۱۱۱۴ تا اضافی می‌شود یعنی ← ۱۰

۲۳۵ ۲۳۸ ۲۴۰

۱۱۱۱۱۱۱۱۱۱ ۱۱۱۱۱۱۱۱۱۵ ۱۱۱۱۱۱۱۱۱۱

۲۳۵ + ۳ + ۲ = ۱۱۱۱۱۱۱۱۱۱۰ ۱۱ ۱۰

۱۲ بیت

در این روش عدد اول را ذخیره می‌کنیم بعد را اختلاف‌ها را ذخیره می‌کنیم در این‌جا چون اختلاف‌ها عددی است، ۳ جای که ۲ بیت فضا اشغال کند ۱۲ بیت فضا اشغال کرده است. اگر شش برابر این که اطلاعات از بین برود) روش‌های فشرده‌سازی وجود دارد اما هر روش یک جا به درد می‌خورد و نمی‌شود یک روش را برای همه‌ی تغییرات در نظر گرفت. مثلاً بهترین روش زمانی است که اختلاف‌ها تغییرات زیادی نکنند و کمترین فضا اشغال کند.

پردازنده‌ی Zilog Z80

این نشان‌دهنده‌ی یک پردازنده‌ی ۸ بیتی است که تقریباً با اختلاف کمی نسل بعد از ۸۰۸۶ آمد که تقریباً مانند هم هستند و فقط یک اختلاف جزئی دارند.

Scanned by CamScanner

در قسمت سمت چپ ساختار داخلی پردازنده را می بینیم (*) و در قسمت سمت راست پایه هایش را می بینیم (**) همانطور که قبلا گفتیم هر پردازنده ای معمولا یک سری پایه دارد و هر پایه برای کاری پیش بینی شده قبل از ورود به ساختار داخلی، ابتدا درباره ی پایه های بحث می کنیم.

این پردازنده یک پردازنده ی ۴۰ پایه است. و برای این که نشان دهیم پایه ی هر IC را نشان دهیم، IC هایی که در طرف استنی آن قرار گرفته اند، پایین از ۱ شروع می شود (آن را در دست چپ ما قرار گیرد) تا ۲۰ و پایه ی شماره ی ۱ در واقع power یا منفی ما است. که برای تغذیه است و پایه ی شماره ی ۴۰ Vcc است. این پردازنده ها ولتاژ ۵ ولت کار می کند البته ولتاژ پردازنده های امروزی بین ۱ تا ۱.۲ ولت است و به مرور کم می شود.

addressbus این پردازنده ۲۰ بیتی است یعنی ۲۰ رودی پایه ی شماره ی ۲ تا A14 نوشته یعنی بیت شماره ۱۴ از addressbus

بعضی از پایه های این پردازنده دو کاره است، مثلا پایه ی شماره ی 38 هم به عنوان A16 یعنی بیت شماره ۱۶ از addressbus و هم برای نشان می دهد هم S3 (status) می باشد یعنی پایه ی دو منظوره.

این پردازنده دو مدل mode (حالت) عملیاتی دارد. می تواند در MAX MODE باشد حالت حداکثری که ما در MAX MODE کار کند. چنانچه می آوریم در هم MODE ای که ما از طریق پایه شماره 33 پایه شماره 33 گفته است. MAX/MIN یعنی اگر به پایه ی 33 ولتاژ ۵ V یعنی یک را وصل کنیم. این در حالت min کار می کند و اگر به پایه ی 33 صفر وصل کنیم در حالت MAX MODE می رود و این اسم هایی که در سمت راست می بینیم نقش پایه ها را نشان می دهد. حتی زمانی که در MAX MODE کار می کند. مثلا فرض می کنیم پایه ی شماره 29 در حالت min پایه ی write است ولی اگر در MAX MODE کار کنیم. همین پایه نقش lock را برای کار دارد. (آن هایی که در پرانتز نوشته شده مربوط به MAX MODE می باشد که بعدا آن ها را بالاسری خط دار یعنی سما فور active هستند)

حال چگونه می آییم MAX یا MIN کار کنیم با توجه به مقدار ولتاژی که به پایه ی 33 اعمال می کنیم و واقع تعیین می کند این پردازنده در حالت maximum کار کند یا minimum.

پایه ی شماره 39 هم در MAX MODE یعنی MAX و در MIN MODE یعنی A15 است. و بعضی از پایه های که در داخل پرانتز نوشته شده است. در واقع نقش آن پایه در حالت MAX را می گوید. در حالت min هم معلوم است یک پایه دوکاره نباشد. که بیشتر به نظر می رسد یعنی دارد یعنی مثلا پایه شماره ۳۶ در یک لحظه A17 و در یک لحظه S4 می شود.

یکی دیگر از تفاوت هایی که این پردازنده با پردازنده های دیگر دارد این است که بعضی از dataBus و addressBus آن ها مشترک است. یعنی پایه ی شماره ۹ را نگاه کنیم. هم پایه شماره ۲ از addressBus و هم پایه شماره ۲ از

Scanned by CamScanner

data bus است که این هم حافظه‌ای است یعنی دریک خانه‌اش address bus و دریک خانه‌اش data bus را نگهداری کند. این‌هایی که به صورت AD0 و AD1 ... می‌باشند دارای دین وتری هستند

* data bus 8086 8 بیتی و data bus 8086 16 بیتی است

چرا 8086 بعد از 8086 به بازار آمد اما علی رغم این که دیسپ‌عدل بعدی up grade می‌شود در این جا مدل قبل را down grade کرد چون قیمت را پایین نیاورد و بازار را حفظ کند.

- شکل پست چیپ را باید بلد باشید که شکل پست راست را فقط نقش پایه‌ها را بدانید (بدون نگاه)

موضوع کتاب شکل

Scanned by CamScanner

جلسه‌ی هفتم
۹۵/۸/۲۲

پردازنده‌ی ۸۰۸۶ و ۸۰۸۸ به‌لحاظ کلی مدل‌های مختلفی دارند. از نظر ساختار داخلی دقیقاً مثل هم هستند هردو ۱۶ بیتی هستند اما تفاوتی که دارند در data bus خارجی آن‌هاست. در ۸۰۸۶ که اول بود data bus ۱۶ بیتی است زمانی که می‌خواهد با حافظه ارتباط برقرار کند به‌شکل ۱۶ بیتی عمل می‌کند هر بار ۱۶ بیت می‌تواند در حافظه بخواند یا بنویسد. به‌شکل ۱۶ بیتی عمل می‌کند. اما در ۸۰۸۸ داخل ۱۶ بیتی است اما می‌خواد با حافظه ارتباط برقرار کند ۸ بیتی ارتباط برقرار می‌کند. البته دقت کنید دلیل اینکه در ۸۰۸۸ علی‌رغم اینکه یکی دو سال بعد از ۸۰۸۶ آمد چرا اکثر ۸ بیتی‌ها؟ ۱- هزینه‌ی ساخت پایین‌تر ۲- توان رقابت. دیباز را بتواند داشته باشد.

تفاوت بین ریز پردازنده و میکروکنترلر:

بعضی موقع‌ها ما با میکروکنترلر و ریزپردازنده میکروکنترلر داریم. چه فرقی دارد این ریزپردازنده و میکروکنترلر؟ میکروکنترلر ما همه‌چی در داخل خودش است. یعنی میکروکنترلر CPU دارد، حافظه درخودش دارد. حافظه Ram و Rum هم می‌توانه داشته باشد. حالا پورت هم در داخل خودش دارد. یعنی یک مینی‌سیستم است برای خودش. البته در حالی که یک ترانزیستور مدرن حالا یک IC.

درصورتی که شما با ریزپردازنده کار می‌کنید اگر بخواهید یک سیستم بسازید خودتون باید برایش حافظه و پورت بزارید و طبیعتاً از نظر ابعاد فیزیکی یک سیستم نرم‌افزاری که با ریزپردازنده ساخته می‌شود فضای بیشتری را اشغال می‌کند تا یک سیستمی که با میکروکنترلر ساخته می‌شود. چرا؟ چون میکروکنترلر همه‌چی درخودش دارد. فنی اینکه راحت‌تر است و سیم‌کشی هم ندارد.

ما در این درس اول ریزپردازنده ۸۰۸۶ رو داریم صحبت می‌کنیم. امروزه استفاده از میکروکنترلرها در پروژه‌های نرم‌افزاری خیلی رایج شده.

شکل ۲ پایه‌های شماره‌ی ۴۰ برای تغذیه پیش‌بینی شده این پردازنده با ولتاژ ۵ ولت کار می‌کند که Vcc یا پایه‌ی شماره‌ی ۴۰ به قطب مثبت ولتاژ ۵ ولت وصل می‌شود و Grand یا زمین به قطب منفی ولتاژ ۵ ولت وصل می‌شود. البته ما دو تا Grand داریم پایه‌های شماره‌ی ۱ و ۲۰ که هردو به قطب منفی یا زمین وصل هستند. برای جلوگیری از نویز از این استفاده می‌کنند.

مثلاً به هر ریزپردازنده‌ای در شکل ۱ addressbus داریم data bus داریم. در شکل ۲ addressbus ها A12 - A13 - A14 ... هستند. این addressbus ها bit هستند. این پردازنده، همان طور که گفتم ۲۰bit است از A_0 دارد تا A19 که می‌شود ۲۰bit و با توجه به این که ۲۰bit است. امکان حافظه‌ای را می‌تواند پشتیبانی کند

خطوط addressbus یا تعداد پایه‌های addressbus ارتباط مستقیم با حجم حافظه‌ای دارد که پردازنده می‌تواند آن را پشتیبانی کند. هرچقدر addressbus بزرگتر باشد یعنی این پردازنده می‌تواند حافظه‌های بزرگتری را بتواند پشتیبانی کند و کار کند.

تعداد پایه آدرس باس
۲ = حجم حافظه‌ای که این پردازنده‌ای که بتواند ببیند.

Scanned by CamScanner

این پردازنده ۲ مدل عملیاتی دارد در دو حالت می‌تواند کار کند یک فول مینیمم است یک مدل مکالمه است که با تراشه به ولتاژی مساوی ۳۳ می‌دهیم یعنی از دنیای خارج به پایه ۳۳ ولت می‌کنیم می‌خواهیم تعیین کنیم که آیا این پردازنده در مد مینیمم کار کند یا مد ماکسیمم

همان طور که می‌بینید در پایه ۳۳ نوشته MN/MX یعنی این پایه است که اگر من به پایه ۳۳ ولتاژ صفر ولت وصل کنم MAX یا ماکسیمم کار می‌کند و اگر به این پایه ولتاژ ۵ ولت وصل کنم MIN یا مینیمم کار می‌کند.

وقتی این پردازنده را در مد ماکسیمم می‌بریم این پردازنده از تراشه‌ی دیگری کمک می‌گیرد مثلاً ۲ تا پردازنده‌ی دیگر

پایه‌ها یک پرانتز باز کرده و یک سری اسامی و توضیحات نوشته شده است. که آن‌هایی که در داخل پرانتز هست نقش اول پایه‌ها است در زمانی که در مد ماکسیمم است. و آن‌هایی که داخل پرانتز نیست نقش اول پایه‌ها را نشان می‌دهد در زمان مینیمم.

مثلاً پایه‌ی شماره‌ی ۲۹ نوشته WR (CPU در داخل حافظه بنویسید) active low است چون بالای آن خط دارد. خاصیت پایه ۲۹ اگر ما در مد ماکسیمم استفاده کنیم این پایه دیگر WR نمی‌شود LOCK که نقش دیگری دارد.

یعنی از پایه‌ها در حالت مد مینیمم هم دو حالته هستند یعنی در یک زمان‌هایی یک نقش دارند در یک زمان‌های دیگر یک نقش دیگر.

مثال: پایه‌ی شماره‌ی ۲۸ در یک لحظاتی به عنوان A16 عمل می‌کند (همان address bus 16 bit) و در یک لحظه‌ای دیگر S3 می‌شود. S5 مقدار پرچم‌های وضعیت هستند و پایه‌ی شماره‌ی ۳۸ دیگر یعنی به مینیمم و ماکسیمم ندارد یا S3 یا A16 است.

درپردازنده برای صرفه‌جویی در تعداد پایه‌ها آمدن بخشی از data bus و address bus مشترک کردند با هم تکیر. درپردازنده‌های نسل قدیم address bus و data bus کاملاً از هم جدا هستند.

ALE سیگنال آمدن بخشی از پایه‌های address bus و data bus را با هم مشترک کرده. به کل این کار می‌گویند AD0 ... AD15 که ۱۶ بیت است که هم data هم address هستند.

مثال: پایه ۱۶ در حافظه‌ی A_0 است و در حافظه‌های D_0 است.

در ۸۰x۸۶ AD از تا ۱۶ بود چون ۱۶ بیت خارجی داشت این جا ۸ بیت.

پایه‌ی شماره ۱۸ و ۱۹ هستند در این پردازنده که برای وقفه‌های سخت افزاری خارجی پیش‌بینی شده‌اند. (INTR و NMI)

فرق پایه‌ی ۱۷ و ۱۸؟

در هر پردازنده‌ای از هر نوع مدلی که می‌خواهد باشد زمانی که یک وقفه اتفاق می‌افتد نرم‌افزاری و یا سخت‌افزاری داخلی و خارجی باشد همیشه اول کار باید یک عددی مشخص شود که آن عدد آدرس مستقیم و یا غیرمستقیم CSR بعد

Scanned by CamScanner

در ۸۰x۸۶ اگر وقفه‌ی سخت‌افزاری خارجی از طریق پایه‌ی شماره‌ی ۱۲ بیاید آن‌وقت عدد ۲ است و اگر وقفه‌ی ما از طریق پایه‌ی ۱۸ بیاید عدد نامعلوم است. CPU منتظر می‌ماند تا عدد را از حیطه‌ی خارج بفرستد. از طریق data bus آن عدد را PIC اعلام می‌کند.

ما می‌توانیم بعضی از وقفه‌ها را غیرفعال کنیم یعنی به CPU بگوییم که اگر وقفه‌ی اتفاق افتاد CPU اتفاقی نیفتد. وقفه‌هایی که می‌توانیم غیرفعال کنیم maskable و وقفه‌هایی که نمی‌توانیم غیرفعال کنیم non maskable هستند. وقفه‌های non maskable هیچ‌وقت غیرفعال نمی‌شوند. هر وقت سیگنال وقفه‌ی شماره‌ی ۲ اعلام شد اگر وقفه‌ای که از پایه‌ی ۱۸ می‌آید از جنس maskable است یعنی شما می‌توانید به CPU بگویید که آن را نادیده بگیرد یا دستور خاص از این قبیل. خود NMI نسبت به INTR اولویت دارد.

پایه‌ی شماره‌ی ۱۹ CLK است. در درس مدار ضمن ما مدارات ترتیبی داشتیم چون مدارات دیجیتال که ساخته می‌شوند دو گروه هستند یا مدار ترکیبی است یا مدار ترتیبی.

خود مدارات ترتیبی به دو شکل ساخته می‌شوند ۱. مدارات ترتیبی سنکرون ۲. مدارات ترتیبی آسنکرون

مدارات ترتیبی سنکرون + مدارهای هستند برای عملکرد و این که مدار بتواند به یک CLK نیاز دارند یا پالس ساعت حال پالس ساعت چیست؟

در واقع یک موج مربعی است که مرتباً بین صفر و یک حال تغییر می‌کند با یک فرکانس مشخص

فرکانس: به تعداد سیکل مدار در واحد زمان فرکانس می‌گوییم.

حال CPU جزو مدار ترتیبی سنکرون است. یعنی همه‌ی CPU ها به CLK نیاز دارند با فرکانس معلوم. و پایه ۱۹ برای موج CLK است.

CLK که وصل می‌شود در بعضی ۸۰x۸۸ ← ۵MHz / ۸۰x۸۶-۲ ← ۸MHz

حال CLK در CPU چه چیزی را مشخص می‌کند. هر چقدر فرکانس CLK بیشتر باشد در یک مدار ترتیبی سرعت اجرای مدار ترتیبی بالاتر است.

سرعت یک CPU به عوامل از جمله سرعت CLK بستگی دارد.

حال این CPU در آن زمان که ساخته شده بود فرکانس در حد MHz بود و الان CPU های فعلی GHz است.

فرکانس CLK در یک CPU محدودیت دارد. یعنی نمی‌توان فرکانس یک CLK را هر چی دلتان خواست بالا برید. نمی‌شد این کار را کرد چون وقتی فرکانس یک CLK را بالا می‌برید مسائلی پیش می‌آید از جمله حرارت. افزایش فرکانس CLK باعث افزایش حرارت می‌شد که مقاومت CPU است.

سرعت یک ریزپردازنده؟

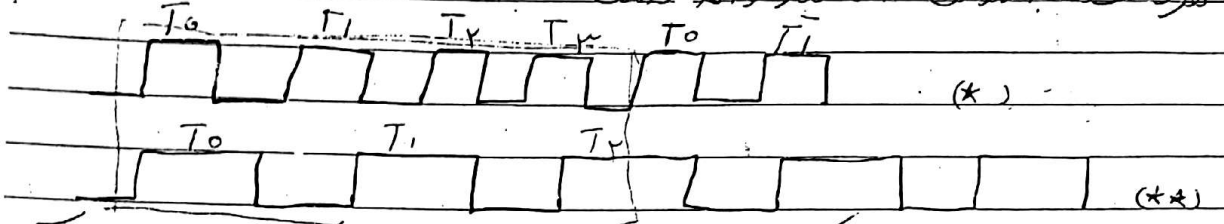
سرعت یک ریزپردازنده یا CPU سرعت به دو عامل بستگی دارد.

۱. فرکانس CLK ۲. معماری خود آن CPU

وقتی CPU می‌خواد یک دستور را اجرا کند این CPU به تعدادی از این پالس ساعت نیاز دارد که به آن CPI می‌گویند. الان داره دستور ADD را مثال اجرا می‌کند.

Scanned by CamScanner

مثلاً در یک CPU فرض دستور ADD به 4 CLK نیاز دارد یعنی CPI = 4 یا MUL دستور
یا CPI = 6 باشد.
پس هر دستوری به تعدادی CLK نیاز خواهد داشت



مثلاً CPU نحوه اجرای دستور ADD را اجرا کند دستور ADD به 4 CLK نیاز دارد و هر کدام یک از این CLK ها یک پالس می‌گیرد یا یک ترتیب می‌آید
یکی از کارهایی که معمار CPU ها بیشتر بالا ببرند به کار کنند CLK و CPI عددی بشه که کمتر شود.
دستور ADD زودتر اجرا شود. CPI را کمتری کند تا دستور سریع‌تر اجرا شود.
اما راه کاری دیگر برای افزایش سرعت CPU می خواهیم راستش به آن است که فرکانس را افزایش بدهیم. این دو شکل موج (*) و (**) این ها فرکانس هایشان با هم فرق دارد. فرکانس (*) بیشتر است.
یک مدت زمان را در نظر می گیریم شکل (*) تا نفری CLK و شکل (**) تا نفری که شکل (**) نوسانش کمتر بوده و شکل * نوسانات بیشتر بوده
و در یک بازه ی زمانی خیلی بیشتر باشد پس فرکانس بالاتر است.
حال فرض کنید که ما به همین CPU که ADD آن به 4 تا CLK نیاز دارد کدام CLK را بدهیم.
اگر CLK (*) را بدهیم تا T3 مدت زمان طول می کشد تا اجرا بشود و اما اگر همین ADD را با CLK (**) بدهیم تا T2 در شکل (**) طول می کشد مدت زمان که طول میکشد تا دستور ADD اجرا شود بیشتر می شود.
هرچند CLK یک CPU را بالا ببریم سرعت اجرای دستورات بیشتر است ولی شرایط دارد.
CPI = clock per instruction
ADD ---> CPI = 4
MUL ---> CPI = 6

over clock
مثلاً clock فلی 3.5 GHz است شما می توانید ببرید 3 یا 4 گیگ.
هر کامپیوتری حق یک برنامه کار بکند clock دارد کار می کند.
حرارت کامپیوتر مشکلات مختلفی دارد یکی فرکانس یا CLK است یکی خود برنامه است.
پایه‌ی مشخصه‌ی CPU 21 پایه‌ی RESET است. همه‌ی CPU ها یک پایه‌ی RESET دارند.
اگر این پایه را تحریک کنیم الآن Activ High است و یک عدد کنیم RESET CPU می شود. الان رخ قرمز روشن کردیم
کنیم یعنی از این پایه ها دو کاره است مثل پایه‌ی 16 همین AD16/S3 است یا ها همین address هست همین data زمان نیاز دارد.

Scanned by CamScanner

مثلاً فرض کنید دستورالعمل (*) ۵ تا T دارد و T۱ و T۲ و T۳ و ... همگی CLK کامپیوتر است. مثلاً مال دستور ADD بوده و T۱ بعدی دارویه دستور دیگر را اجرا کند. هر دستوری تعداد پالس ساعت ی خواهد مثلاً دستور ADD همیشه تا T۳ ی را میخواهد مثلاً دستور ضرب تا T۵ ی را میخواهد. برای همین دستورات اوایل پایه‌های AD که دوکاره هستند مثلاً تا T۳ address هستند از T۳ به بعد data می‌شوند. مثلاً یک دستور دارد اجرا میشود در T۱ و T۲ روی پایه‌ها address هستند از T۳ به بعد data می‌شوند پس در حین اجرا در یک زمان یک نقش و در زمان دیگر یک نقش دیگر دارند.

پایه‌ی ۳۱ و ۳۰ برای عمل DMA داریم.

هایک DMA ، request داشتیم یعنی او دستگاهی که می‌خواست دیتا حجم انبوهی از اطلاعات را بفرستد DMA request می‌زد به CPU و AddressBus و databus خودرا آشنا می‌کرد. یعنی به جای اینکه اطلاعات از Port یا دیتا از CPU عبور کند مستقیم دارد از حافظه از control DMA را دور می‌زند و به حافظه می‌رفت.

حال در ۸۰x۸۶ هم ما همین پایه را داریم فقط اسمش را عوض کرده به جای DMA request گذاشته HOLD به جای آن DAM Acknowledge گذاشته HOLD Acknowledge (HLDA).

HOLD آمد از دنیای خارج بخواد DAM بزنه این پایه را فعالش می‌کند HLDA همون پایه‌ی خروجی است یعنی CPU می‌خواد اطلاع بده که می‌خواهد اون عمل DAM هست را انجام دهد این پایه را فعال می‌کند.

در هر سیستم ریزپردازنده‌ای ۴ تا پایه داریم یا پایه‌های ورودی هستند یا پایه‌های خروجی هستند یا پایه‌های دو طرفه هستند.

- پایه‌های ورودی پایه‌هایی هستند که اختیار صفر یک کردن دست خودشان نیست و از دنیای خارج صفر و یک می‌شنود. مثل پایه‌ی ۳۲ یا پایه‌ی HOLD ۸۰۸۶ ۵ مدت یک ۰ میشنود
- پایه‌های خروجی اختیار صفر و یک کردن دست CPU است HLDA
- آن پایه‌هایی که دو طرفه هستند بعضی وقت دست CPU هست و بعضی وقت هایست.

پایه‌ی شماره‌ی ۳۲ RD active low هست همین CPU هایی پایه‌ی RD دارند. زمانی که می‌خواهد از پورت یا حافظه بخواند آن را فعال می‌کند. البته بعضی از CPU ها برای خواندن از پورت به جای RD از IORD استفاده می‌کنند.

زمانی که بخواد بنویسد از پایه‌ی WR استفاده می‌کند.

این CPU ها پردازنده‌ای برای اینکه بتوانیم تفاوت قائل بشیم برای خواندن و نوشتن در حافظه خواندن و نوشتن در پورت‌ها همین طور که می‌بینید پایه IORD و IOWR نداره اما آمده یک پایه‌ی ۲۸ درست کرده به نام M/IO که خروجی است. آن CPU بخواد از حافظه بخواند پایه‌ی M/IO را صفر می‌کند و اگر CPU بخواد از پورت بخواند و بنویسد آن پایه‌ی ۲۸ را یک می‌کند.

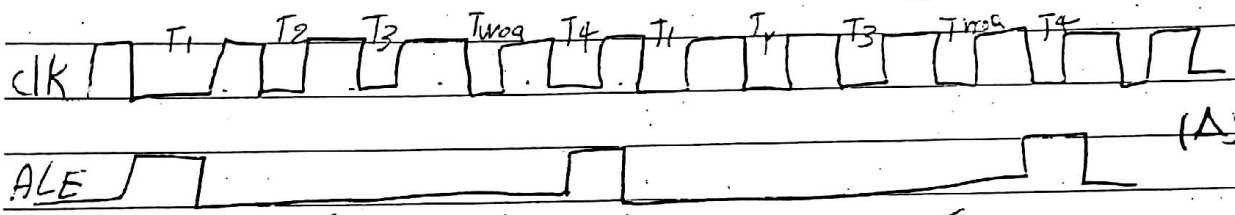
Scanned by CamScanner

پایه های ۲ پایه های اینتوس هستند وضعیت رادیک لاچ خارجی می دهند با نقاط کردن به آدرس های نقطه CPU که در چه وضعیتی است. در چه حالتی از امرای دستور است.

پایه ی DT/R مخفف Data transmit Receive . این پایه ی خروجی است زمانی که CPU در حال ارسال اطلاعات باشد یک است در عکس CPU در حال دریافت اطلاعات باشد صفر است.

پایه ی DEN مخفف Data Enable : Data bus در یک زمانی مقداردهی می شود زمانی که data مقدارش valid گذاشته می شود CPU می آید زمانی که اطلاعات معتبری روی خطوط databus قرار می گیرد Data Enable فعال می کند.

پایه ی ۲۵ پایه ی ALE : مخفف Address Lach Enable . این پایه یک کاربرد خیلی مهم دارد. به کمک این پایه ما می توانیم address و data را از هم جدا کنیم. در این صورت CPU ما address و data مالتی پلکس شده یعنی مشترک هستند.



(۵)

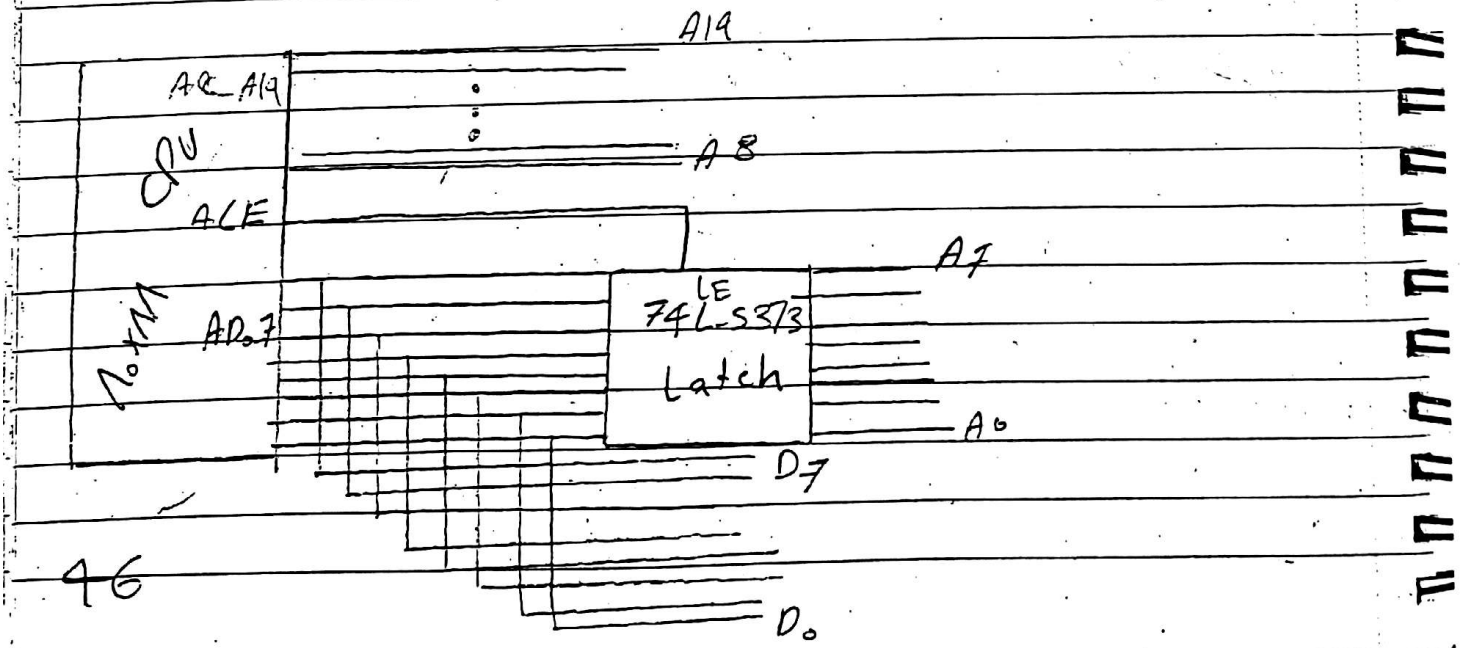
در ابتدای فردستور یعنی زمانی که CPU می آید ALE یک لاچ خارجی را فعال می کند.

پایه ی شماره ی ۲۶ یک پایه ی خروجی است. INTA که active low هم هست

INTA = Interrupt Acknowlede

زمانی که CPU وقفه می خورد وقتی CPU می رود تا به وقفه رسیدگی کند INTA را صدا می کند یعنی در رسیدگی ارتباط عادی که CPU با کارت عادی دارد یک است. چراغ می کند یعنی سخت افزارهای دیگر به من CPU در حال رسیدگی به وقفه است.

چه موقع می خواهیم در پردازنده های ۸۰x۸۶ address و data را از هم جدا کنیم؟ وقتی که address و data



Scanned by CamScanner

وقتی یک CPU می خواهد یک دستور را اجرا کند در سیکل یا پالس اول یا T_1 یا T_1 این پایه ها همگی نقش address را دارند یعنی AD_7 تا AD_0 نقش address را دارند و بقیه نقش آدرس را دارند و از پالس دوم به بعد این پایه ها تغییر وضعیت می دهند یعنی دیگر AD_7 تا AD_0 address نیست بلکه data می شود

ALE

حال برای هر بار که یک دستوری خواهد اجرا شود باید این پالس اول را بدهیم تا همه address ها را داشته باشیم. در این میکروکنترلرها می آیند address ها را در ثبات هایی ذخیره می کنند که به آن Latch می گویند. همان اول کار در ثباتی ذخیره می کنند که وقتی تغییر وضعیت داد یعنی address دیگر نیست باید درشان AD_0 تا AD_7 و همان لحظه که AD_0 تا AD_7 آدرس را داشتند باید Latch ها عمل کنند و ALE عمل است

Latch ها چه زمانی data را در خودش ذخیره می کند؟ زمانی که فعال باشند

هر دستوری خواهد اجرا شود مطابق شکل (A) در اولین Clk، ALE یک می شود یعنی فعال می شود اول ALE که Latch عمل است در T_1 یک است و در Latch بازی می کند و AD_7 تا AD_0 که address هستند را در خودش ذخیره می کند و بعد ALE می رود در Latch بسته می شود بعد AD_7 تا AD_0 می رود data و دیگر وارد Latch نمی شود حال ALE در رابطه وقتی قرار دارد عمل می کند. با این روش address و data را جدا کردیم.

این روش در میکروکنترلر ATmega عمل استفاده شده است.

47

Scanned by CamScanner

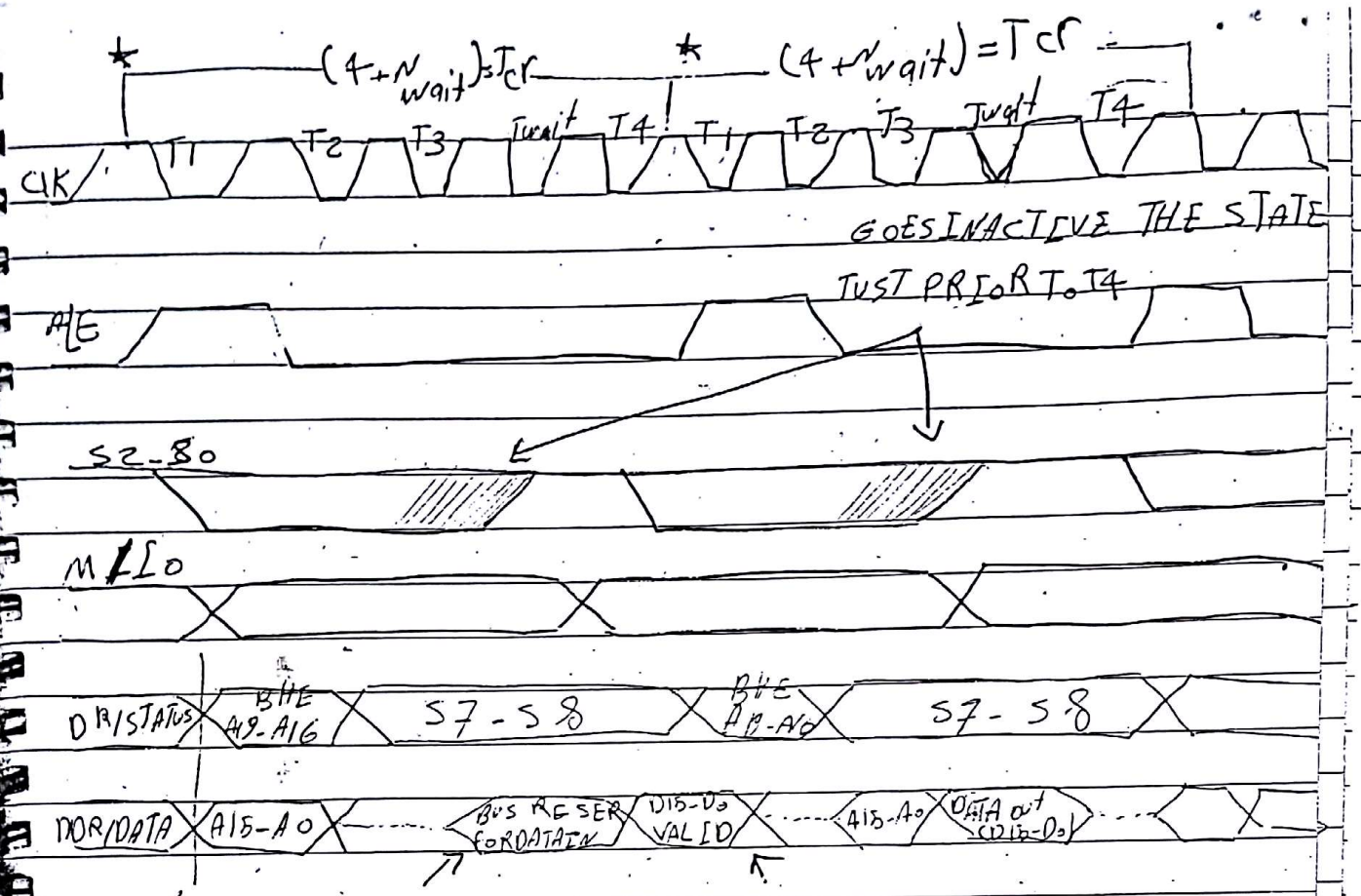
جلسه‌ی نهم:

پردازنده‌ی 8086 و 8088 هردو از لحاظ عملکرد داخلی کاملاً یکسان اند. هردو پردازنده‌های 16 بیتی هستند، ساختار داخلی در هر 2 مطابق شکل سمت چپ است (شکل پردازنده 8088) اما تفاوت‌های آن‌ها این است که در پردازنده‌ی 8086 خارجی 16 بیت می‌باشد یعنی زمانی که سراغ حافظه یا سراغ Port ها برویم، 16 بیت می‌باشد. اما در داخل دو هردو 16 بیت و address bus هم چند بیت می‌باشد؟ 2^{20} یا 1MB حافظه را می‌تواند شناسایی کند.

با یه سنگاره چی؟ Ready نگه دارد. یعنی وقت‌هایی که می‌خواهیم CPU و حافظه را به هم وصل می‌کنیم با سرعت CPU هم‌خوانی نداشته باشد. مثلاً فرض می‌کنیم دستور ADD به 4 CLK (پالس ساعت) نیاز دارد. تا انجام شود حال این دستور ADD قطعاً ممکن است نیاز به مراجعه به حافظه هم داشته باشد. مثلاً می‌خواهیم عدد درون یا محتوای خانه‌ی 4 حافظه ADD کند یعنی باید محتوای خانه‌ی 4 حافظه را بردارد تا با چیز جمع کند و جواب را دوباره در آکومولاتور ذخیره کند. یعنی ما در این دستور نقل و انتقال با حافظه داریم. اگر سرعت حافظه کافی نباشد، ممکن است بیش از 4 تا CLK نیاز داشته باشیم. چون ممکن است زمانی که CPU سراغ حافظه می‌رود، حافظه به هیچ‌وجه سریع جواب ندهد. مثلاً وقتی می‌خواهیم خانه‌ی 4 را به مابقه حافظه CPU را معطل کند. یعنی CPU ای که به صورت 4 در نظر گرفتیم و دستورالعمل این‌گونه است سرعت حافظه پاسخ‌گو باشد و با سرعت CPU هم‌خوانی نداشته باشد. حال برای رفع این مشکل باید چه کرد؟ چه کار کنیم که این حافظه‌ی کند (یا مثلاً RAM هم‌چنین) با CPU تبادل اطلاعات داشته باشد؟ از دو چنین مواقعی ما می‌توانیم از wait state استفاده کنیم یعنی CPU می‌تواند حالتی داشته باشد به نام wait state یعنی حالت انتظار که در این حالت CPU به اندازه‌ی 1 CLK bus صبر می‌کند و فعلیت پایه‌های خودش را حفظ می‌کند تا حافظه آماده برای ارسال جواب دهد. حالا به اندازه‌ی 1 CLK bus یا 2 CLK bus چی؟ که به این wait state می‌گویند.

باید حواسمان به شکل زیر باشد که گفتیم که ما دیدیم یک دستور را مانند دستور جمع یا ضرب به تعداد پالس ساعت نیاز دارد که در هریک از این پالس‌ها اتفاقاتی می‌افتد تا دستور کامل شود. حالا این حالات wait state داشته باشیم. CPU در آخر CLK bus، S که به عنوان wait state در نظر گرفتیم عملکردی انجام می‌دهد یعنی عرضه‌ی داده را انجام می‌دهد و وضعیت خود را حفظ می‌کند.

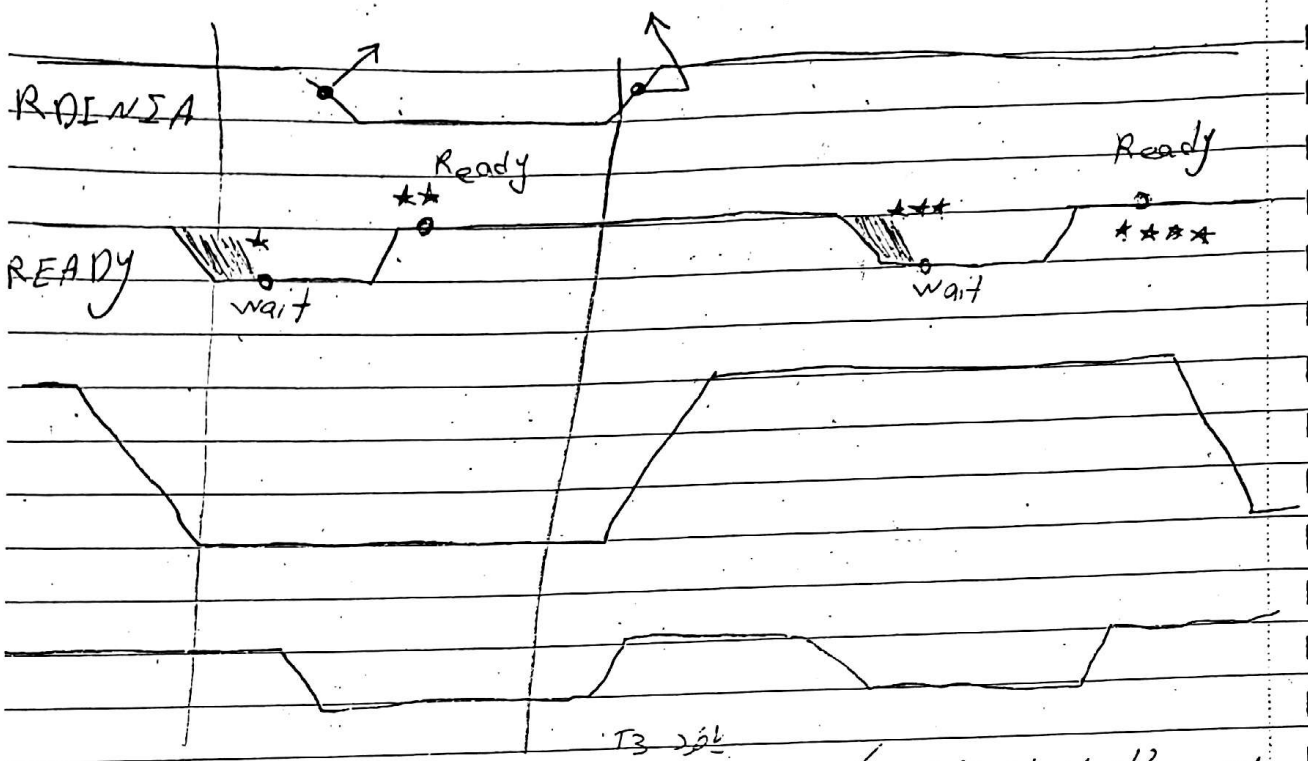
Scanned by CamScanner



در شکل بالا از * تا * یک دستور است. در حالتی که T3 چون نیاز به حافظه داریم و حافظه آماده در CLK بعدی داده در T4 می‌دهد به جای T4 می‌نویسیم Twait یعنی وارد wait state می‌کنیم یعنی CPU به اندازه‌ی یک پالس ساعت صبر می‌کند و کاری انجام نمی‌دهد (در حالت wait state) CPU مقادیر خاصی را به ماژول‌ها می‌دهد و آنها را حفظ می‌کند هیچ وابستگی تغییر وضعیت نمی‌دهد) و بعد از آن حالت wait وارد فاز بعدی اجرای دستور می‌شویم یعنی T4 های که به دلیل کند بودن حافظه اضافه کردیم منظور به افزایش زمان اجرای دستور می‌شود به اندازه‌ی هر wait state یک CLK bus زمان اجرای دستور را اضافه می‌شود هر چه تعداد wait state ای نداشته باشیم یعنی سرعت حافظه‌ی ما بالا باشد به طوری که نیاز به wait state نباشد. خیلی بهتر و سریع‌تر)

سوال: CPU از کجا بفهمد که حافظه برای دادن داده برای خروج از Wait state نیاز دارد؟
به وسیله‌ی سیگنال READY این سیگنال یک سیگنال ورودی است. CPU در یک مرحله‌ای که نیاز به حافظه داشته برای اینکه بفهمد wait state چه کند یا نه سیگنال Ready را چک می‌کند. اگر 0 = Ready باشد فاز بعدی به wait state می‌رود اما اگر برابر با 1 باشد یعنی حافظه آماده می‌باشد پس نیازی به wait state نداریم.

Scanned by CamScanner



باقی به شکل بالا دقت کنید * مربوط به T2 و T3 می باشد پس فاصله ی بین آن دو را Ready چک کردیم که برابر با صفر شده چون Ready برابر بود با عدد یک پس رفت به Wait T3 داشته باشیم یعنی باید یک CLK صبر کند تا حافظه آماده شود. در این Wait state دوباره Ready با درنقطه ی ** چک می کنیم که زمانی را نشان می دهد که در حالت wait بوده و چون Ready در نقطه ی *** برابر با یک بود پس یعنی حافظه آماده شده بنابراین بعد از Wait دیگر Wait نداریم و وارد مرحله ی T4 شدیم. باز مثلاً در Ready *** را چک کردیم که مربوط به T4 می باشد چون Ready برابر صفر بود پس بعد از T3 دوباره wait داریم. داخل wait state در نقطه *** دوباره Ready را چک کردیم و چون برابر با یک بود دیگر به CLK های نیاز به wait state نیست و وارد مرحله ی T4 شدیم.

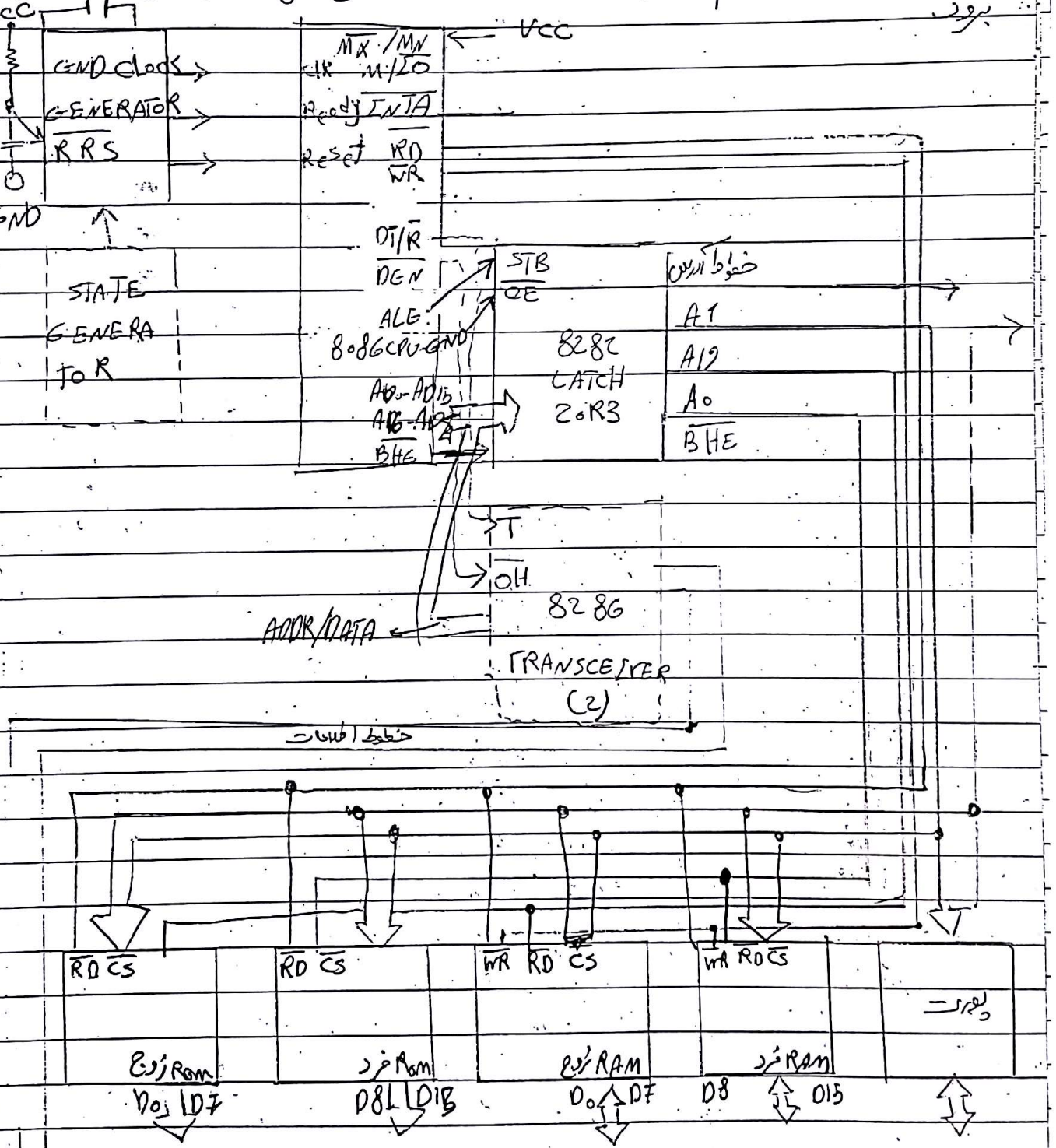
نکته ی مهم این است که Ready را از حافظه می گیرد برای مادرست می شد

* ما در جلسات اول نحوه ی ارتباط CPU با حافظه را دیدیم و گفتیم یک address bus و data bus و Read و Write داریم. و از طریق این ها ما CPU به حافظه وصل می کردیم حالا می خواهیم این ارتباط را در یک نمونه ی عینی و واقعی ببینیم یعنی می خواهیم این ارتباط ها را در یک نمونه ی 8086 بررسی کنیم

در شکل صفحه ی بعد یک پردازنده ی 8086 را می بینیم که به حافظه وصل شد. همه ی حافظه های ما در سیستم ما وصل می کنیم همگی است حافظه ی ما با CPU سازگار باشد و برعکس وقتی وقت ما در سیستم ما ما از چندین ماژول حافظه استفاده می کنیم.

Scanned by CamScanner

تا CPU را با حافظه ها مثلا کنیم مثلا در طراحی 8086 دیدیم که address bus ۲۰ بیتی است و 2^{20} یا 1MB حافظه را می تواند شناسایی کند. اگر ماژول حافظه 512K یا مثلا با ید دوتا ماژول 512K به شکل هم استفاده کنیم تا بتوانیم 1MB را فراهم کنیم و CPU بتواند کار کند یا مثلا ۲ ماژول حافظه 256K باشد. باید حتما ماژول حافظه بگذاریم یا مثلا data bus مربوط به CPU ما ۱۶ بیتی باشد از طرفی data bus مربوط به ماژول های حافظه ۸ بیتی باشد پس باید دوتا دوتا سازی کنار هم بچینیم یعنی ۸ بیت کم ارزش تر یک ماژول و ۸ بیت پر ارزش تر ماژول دیگر بروند.



Scanned by CamScanner

بس سیستم به نوع module هایی که قرار است استفاده کنیم بعضی اتصالات ی تواند متفاوت باشد
این سئوال صفحه ی قبل برای شما این بعضی اتصالات حل می باشد که از چه ماژول هایی
استفاده کرده یک سری از این module ها برای آدرس های زوج و یک سری دیگر برای آدرس های
فرد است یعنی مثلاً آدرس های خانه هایی که انتخاب کردیم عدد فرد یا شد ماژول های فرد فعال
شده و اطلاعات دیتا آن شد و یا از آن ها خوانده می شود و برای زوج هم همین طور. از طرف
هر یک از module های حافظه ی فقط است اطلاعات می توان بگیرد یعنی databus آن
پنج است. در صورتی که databus مربوط به CPU ما 16 بیتی است.

در 8086 پایه ای به نام BHE (Bus High Enable) داریم و پایه ی متناظر آن در 8088 و 80286 و 80386 و 80486 و 34 ی باشد.
یا وقتی پایه A0 کم ارزش وضعیت است.

در صورتی که در 8088 چنین پایه ای نداریم. پایه ی SSO در 8088 و 80286 و 80386 و 80486 و 34 ی باشد.

حال با توجه به سئوال صفحه ی قبل می دانیم که در یک سیستم ریز پردازنده به حافظه ی RAM نیاز داریم و یا توجه به این سیستم که یک منتور ما چه منتظر ماست به حافظه RAM هم ممکن است نیاز داشته باشیم.
بنابراین در این سیستم هم ROM و هم RAM داریم ROM که ضروری است و RAM هم داریم
در این سئوال تقسیم بندی این گونه انجام شده است. خانه هایی که آدرس فرد دارند هم ROM و هم RAM)
یک module برای خانه های فرد ROM و یک module برای خانه های فرد RAM داریم. خانه های فرد
از خطوط D8 - D15 استفاده می کند یعنی databus پر ارزش (high) از فرد ها استفاده می شود
و تا module هم برای ROM و RAM زوج داریم که مطابق سئوال از D0 - D7 استفاده می شود.
پس کلیه اطلاعات را از خانه ی زوج بخواهیم به صورت 1 بایتی از خطوط databus D0 - D7 استفاده
می شود و اگر بخواهیم 2 بایت از خانه ی فرد بخوانیم از خطوط D8 - D15 استفاده می شود و اگر بخواهیم
بایت بخوانیم یا یک خانه ی فرد و یک خانه ی زوج یا هم خوانده شود.

حال بعضی اتصالات به چه صورتی است و مطابق سئوال چون address bus و data bus مشترک
هستند Latch را از یک سطوحی 373 استفاده می کنیم و پایه ی OE متناظر 373 و 573 و
پایه ی LE ورودی با نام STB (strobe) داریم که اگر این ورودی strobe پر باشد
یا 1 باشد اطلاعات ورودی Latch ذخیره می شود و اگر strobe برابر با صفر
یا 0 باشد اطلاعاتی داخل آن نمی رود در این حالت ورودی STB به خروجی ALE وصل شود
یعنی زمانی که آدرس روی address bus و ALE برابر با 1 است. اطلاعات
داخل Latch ذخیره می شود و وقتی ALE برابر با صفر شد دیگر ورودی Latch مسدود
شده چیزی داخل آن نمی رود و خطوط تغییر وضعیت داده و address به data تبدیل
می شود. خروجی این Latch ها A0 - A19 می باشد در این حالت AD ها جدا کرده
مطابق سئوال module هایی که از جنس RAM هستند (چه زوج و چه فرد) فقط سیگنال
RD به کدام یک متصل شده همیشه خط RD مربوط به CPU به RD مربوط به حافظه
باید وصل باشد. و خط WR هم همین طور) خط WR را دارد چون RAM است اما

Scanned by CamScanner

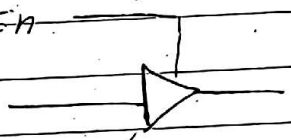
تبعیت ها به صورت مربوط به RAM می باشد. خط WR هم دارد.
معمولاً ماژول های حافظه یک پایه دیگری به نام CS (Chip select) دارند.
هر ماژول RAM زمانی قابل استفاده است که CS آن فعال باشد (این ها active low است)
پس برای هر ماژول حافظه یک CS داریم که معمولاً active low می باشد یعنی اگر CS برابر با صفر شد
ماژول حافظه فعال خواهد بود و اگر CS برابر با 1 باشد ماژول حافظه غیرفعال خواهد بود. برای ماژول
های زوج و فرد چگونه کنترل می کنیم؟ CS ماژول زوج به CS ماژول زوج دیگر وصل است و برای فرد
هم همین طور. حالا CS ماژول زوج به A_0 وصل شده یعنی هر وقت $A_0 = 0$ باشد ماژول های
زوج فعال می شوند. چون اعداد مبنای ۲ اگر زوج باشند سمت راست ترین بیت آن ها صفر و اگر فرد
باشد یا می باشد. A_0 سمت راست ترین بیت می باشد) از طرفی CS ماژول های فرد به
BHE وصل شده که به دلیل طبیعت پردازنده ی 8086 می باشد. به جدول زیر توجه کنید

BHE	A_0	نوع انتقال
0	0	انتقال اطلاعات از طریق 16 بیتی (کلمه ای)
0	1	انتقال اطلاعات از طریق خطوط با بیت بالا (D15-D8) آدرس های فرد
1	0	انتقال اطلاعات از طریق خطوط با بیت پایین (D7-D0) آدرس های زوج
1	1	هیچ عملی صورت نمی گیرد (idle)

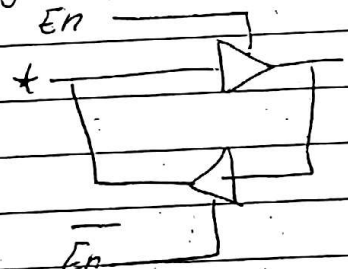
با توجه به جدول می توان دو سیگنال های BHE و A_0 را مشخص.
اگر $A_0 = 0$ ماژول های زوج انتخاب شده و خطوط D7-D0 فعال و قابل استفاده است.
از طرفی اگر BHE = 0 ماژول های فرد نیز فعال شده و D15-D8 از خطوط data bus انتقال می کند
بنابراین بایست از data bus دو تا بیتم استفاده کنیم.
اگر $A_0 = 0$ باشد یعنی آدرس فرد است و از طرفی BHE همواره A_0 است پس
ماژول های زوج انتخاب نمی شوند و BHE = 0 است پس ماژول های فرد انتخاب می شوند.
در نتیجه خطوط D15-D8 فعال شده و در این حالت فقط 8 bit ای می توانیم بخوانیم.
یعنی اگر از یک آدرس فرد بخواهیم فقط 8 bit ای می توانیم بخوانیم.
اگر $A_0 = 0$ باشد یعنی آدرس زوج است پس ماژول های زوج یعنی D7-D0 فعال اند اما BHE = 1
می باشد یعنی ماژول های فرد غیرفعال هست در نتیجه فقط 8 bit ای می توانیم بخوانیم که انتقال اطلاعات
از خطوط D7-D0 می تواند انجام شود.
اگر $A_0 = 1$ و BHE = 1 باشد هیچ عملی صورت نمی گیرد

Scanned by CamScanner

* جدول صفحه‌ی قبل می‌گوید: اگر ما بخواهیم ۱۶ بیت اطلاعات را از RAM بخوانیم و یا حداکثر سرعت ممکن باید از خانه‌ی زوج بخوانیم که تمام خطوط Databus فعال است (حالت اول)
اگر بخواهیم ۱۶ بیت اطلاعات را از یک خانه‌ی فرد بخوانیم یا باید ۲ بار سراغ حافظه برویم. چون وقتی سراغ خانه‌ی فردی می‌رویم فقط یک بایت می‌توانیم بخوانیم. اگر دو بایت بخوانیم یعنی یک خانه‌ی فرد و یک خانه‌ی زوج. پیشت سرش پس یک بار باید ۱ خانه‌ی فرد و یک بار خانه‌ی زوج را بخوانیم یعنی انتقال ۱۶ بیتی از خانه‌های زوج و سرعتش ۲ برابر انتقال ۱۶ بیتی از خانه‌های فرد است. پس برای برنامه‌نویس باید سعی کنیم آدرس زوج باشد که با یک سیگنال کار تمام شود.
حالا به سؤال قبل باز می‌گردیم و می‌بینیم که latch در address ذخیره شده و data وارد CPU شده و وارد یک Transceiver شده. Transceiver چیست؟
حالا در درس مدار منطقی که خواندیم که بافر ایمانی است با سمبل ▷ که ورودی و خروجی یکسانی دارد. این بافر یک بافر معمولی است که برای تقویت جریان از آن استفاده می‌کنیم و جهت بافر مهم است که کدام طرف ورودی و کدام طرف خروجی است زیرا اگر اشتباه وصل کنیم بافر می‌سوزد.
نوعی بافر دیگر به نام بافر سه حالته یا Tri state نیز داریم که سمبل آن به صورت زیر می‌باشد و ورودی Enable هم دارد.



اگر ورودی En فعال باشد یعنی En=1 باشد، ورودی در خروجی یکسان خواهد بود پس یک بافر معمولی داریم اما اگر En=0 باشد خروجی قطع می‌شود و به این حالت Float یا شناوری گویند (High ambalance) حالت لیست از Transceiver از دو بافر ۳ حالته مطابق شکل زیر به وجود آمده:



یعنی خروجی یکی به ورودی دیگری و برعکس است خروجی دیگری هم به ورودی دیگری. اما هنوز تعیین نمی‌کنیم که از کدام طرف به کدام طرف باشد. En داریم یا نه En داریم. حالا می‌خواهیم ببینیم چگونه کار می‌کند اگر En را 1 بدهیم بالایی فعال و پایینی قطع است. چون En برابر صفر می‌شود حالا اگر اطلاعاتی از ورودی * بیاید اطلاعات از طریق بافر به خروجی انتقال پیدا می‌کند اما اگر اطلاعات از ** بیاید اطلاعات نمی‌تواند عبور کند و ورودی سمت راست هم دومی یعنی En خاموش است بین این En=1 باشد جهت اطلاعات از چپ به راست می‌باشد حال اگر En را 0 بدهیم بافر بالایی قطع و بافر پایینی وصل است یعنی اگر اطلاعاتی از سمت راست بافر بالایی عبور کردیم از طریق بافر پایینی اطلاعات به سمت چپ منتقل می‌شود ولی اگر اطلاعاتی از سمت چپ بیاید نمی‌تواند به سمت راست بیاید چون بافر بالایی خاموش است و در بافر پایینی انتقال اطلاعات از چپ به راست ممکن است

Scanned by CamScanner

سی در این حالت فقط از راست به چپ می تواند برود یعنی به کمک کنترل En می توانیم یکسو جهتی اطلاعات از چپ به راست باشد یا از راست به چپ. Trasceiver ها

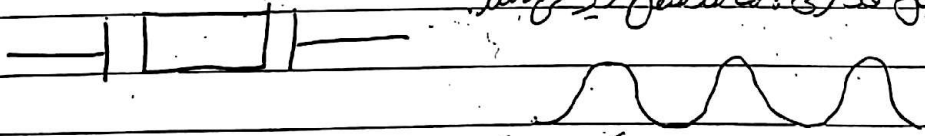
دقیقاً چنین کاری را انجام می دهد. درداخل Trasceiver بستگی به چند بیت بودن آن دارد در این جا 8 بیت است. حتماً اینرا Transceiver 8 بیتی داشته باشید. A تا I از شکل I داخل آن موجود می باشد که A و H همه به هم وصل است و En هاهم همین طور که می توانیم یکسو اطلاعات از چپ به راست یا راست به چپ برود. حالا در شکل همه خطوط address/data را داخل Trasceiver آوردیم که 16 بیتی است. این Trasceiver یک واسطه ی T (Trasceiver) دارد به DT/R وصل است که اگر این پایه یک شود داده ها می روند یعنی اطلاعات از چپ به راست می رود) یا بالعکس یعنی اطلاعات از چپ به راست می توانند برود. طرف خط T برابر با صفر باشد. Tranceiver از راست به چپ عمل می کند داده ها از دنیای خارج وارد CPU می شوند پس با Tranceiver جهت اطلاعات را می توانیم مشخص کنیم. از طرفی خود Tranceiver یک پایه En یا OE (Output Enable) دارد که به DEN وصل است. Transceiver چه وقت روشن می شود؟ زمانی که data و address/data با سیگنال Enable یک شود. در واقع ما گفتیم زمانی که خطوط address/data نقش data را دارند. DEN همزاد است که یعنی فعال است و اگر خطوط address/data نقش address را داشته باشند. DEN تریتا یا است پس اگر DEN نباشد یا صفر باشد Transceiver کار نمی کند. درورودی های خروجی ها به هم وصل نیستند. داخل Transceiver از دو تا databus تشکیل شده است که خطوط databus کاملاً جدا هستند.

قسمت D0 - D7 خطوط ارتباط به module های مربوط است و A0-A7 به module های خروجی وصل شده. Port ها همه که به خطوط address وصل است در واقع همه از طریق Port ها می توانند داده را از CPU بگیرند (ورودی) و همچنین می توانند به بیرون بروند. پس Port ها به address bus وصل است که به کمک address bus ها می توانیم Port مورد نظر را انتخاب کنیم. البته در I/O هم باید به گونه ای دخیل شود که فاصله های حافظه ای Port خالی نشود.

* در پروژه زنده ی 8086 به جای 8088 و M/IO، IO/M داریم که عکس آن هاست برعکس است.

معمولاً CPU ها به یک Clk Generator نیاز دارند. Clk Generator برای IC است. Clk Generator ها را تولید کننده ی پالس ساعت چه کاری کند؟

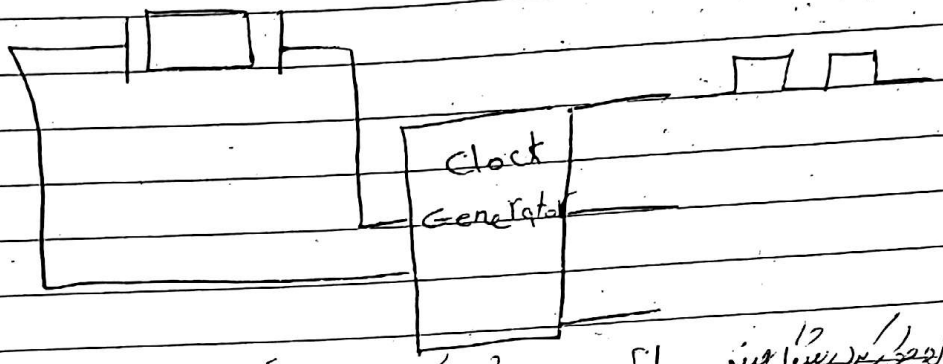
این Clk Generator ها با یک کریستال وصل می شوند. پس ما یک قطعه ای داریم به نام کریستال که از تغییر سیلیکون ساخته شده است.



کریستال یک قطعه یا device می باشد که با یک موج سینوسی با فرکانس و دامنه ی مشخص می زند. (ورودی) حال ما می دانیم که Clk مربوط به CPU به موج سینوس نیاز ندارد بلکه به موج مربعی نیاز دارد. بنابراین از یک تراشه ی Clk Generator استفاده می کنیم که

Scanned by CamScanner

حالا خروجی کریستال را به Clk Generator وصل می‌کنیم و درنتیجه سیگنال موج مربعی سینوس را به چندین موج مربعی با فرکانس‌های متفاوت تبدیل می‌کند. البته این فرکانس‌ها همگی از مضربی از عددی هستند مثلاً اگر فرض کنیم کریستال 14MHz باشد خروجی Clk Generator ما 14MHz و 7MHz و 3.5MHz و ... خواهد بود. تقسیم بر یک عددی می‌کنیم و به سیگنال موج مربعی که داریم تبدیل می‌شود (با فرکانس‌های مختلف)

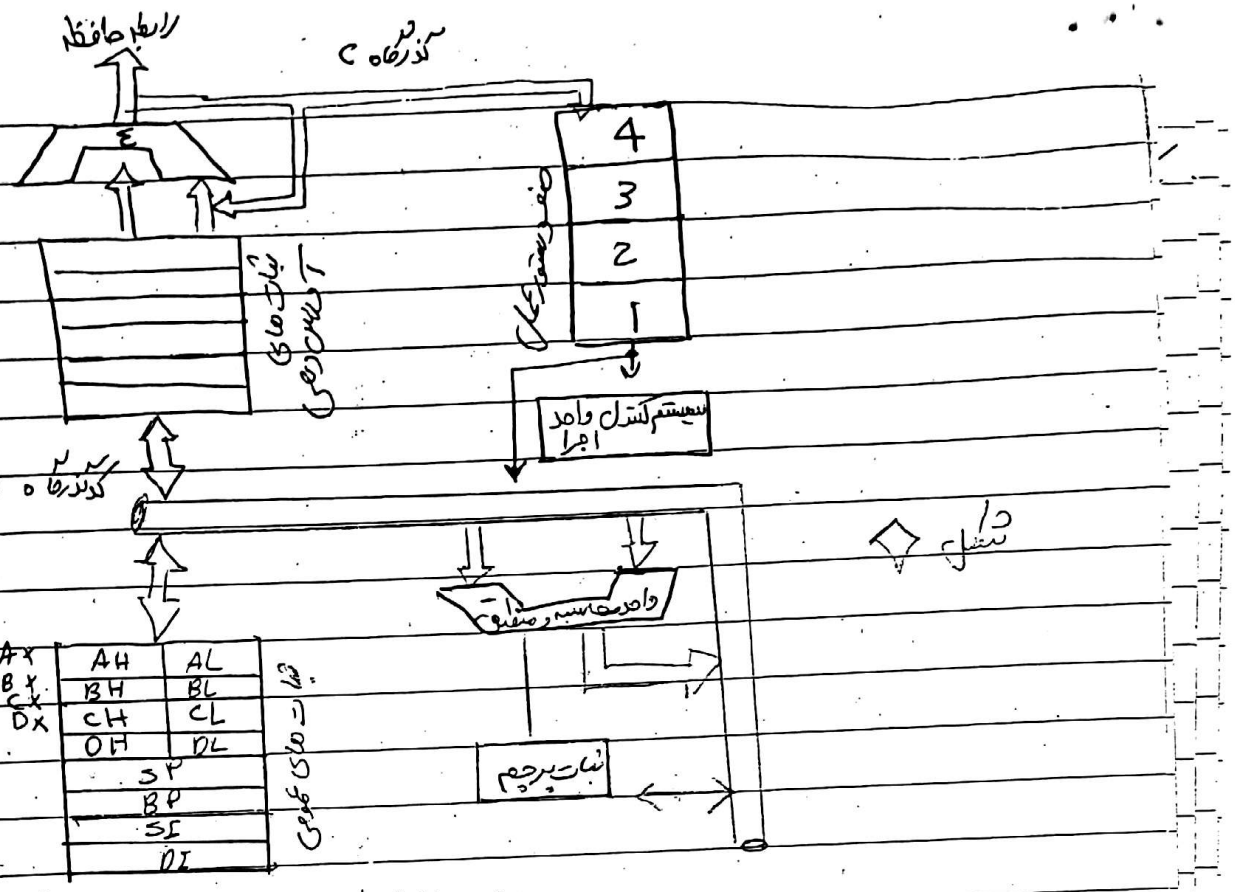


همانطور که در شکل هفتم آمد خروجی‌های Clock generator به ورودی CPU CLK متصل شده است. یعنی سیگنال موج مربعی که برای کارکرد CPU لازم داریم از این Clock generator می‌گیرد. همانطور که گفتیم برخی CPU ها مثل 8086 که ما داریم CLK آن 5MHz بود که در برد ما این فرکانس در نظر گرفته شده است. برخی CPU ها نیز فرکانس بیشتری می‌خواهند. Clock generator نیز علاوه بر CLK سیگنال RES را نیز تولید می‌کند. دلیل این که Clock generator این سیگنال را نیز تولید می‌کند این است که ممکن است در یک زمان اتفاق بیفتد که ما بخواهیم سیستم را Reset کنیم. برای همین RES را به Clock generator وصل می‌کنیم و خروجی Reset هم دارد که این خروجی به ورودی Reset CPU وصل می‌شود. همچنین ما یک ورودی داریم به نام Ready که به wait state Generator وصل می‌شود. wait state Generator نیز خروجی‌اش Ready است که آن را به ورودی Ready مربوط به CPU وصل می‌کنیم. پس Ready و RES را از داخل Clock generator بیرون می‌دهیم تا به مشکل برنخوریم. می‌توانستیم مستقیم همین کار را انجام دهیم اما این گونه عمل کردیم تا بتوانیم CLK را به سیگنال‌های دیگر نیز متصل کنیم. علاوه بر این که سیگنال CLK را به سیگنال‌های دیگر نیز متصل کنیم. بعضی از وسایل جانبی را با CLK مشکل پیدا می‌کند.

این سیگنال‌ها معمولی بود. ساختار داخلی این پردازنده‌ها 8086، 8088 و 80286 را نشان می‌دهد. داخل آن‌ها چه چیزهایی داریم.

همیشه زمانی که می‌خواهیم با یک پردازنده کار کنیم اولین چیزی که باید بررسی کنیم وضعیت ثبات‌های آن پردازنده است و همین پردازنده‌ها داخل خودشان یک سری ثبات دارند که برای انجام عملیات داخلی خودشان از این ثبات‌ها استفاده می‌کنند. برنامه‌نویس‌ها هم برای نوشتن برنامه‌هایشان می‌توانند از این ثبات‌ها استفاده کنند.

Scanned by CamScanner



معماری داخلی پردازنده 8088

ما در زبان C دستوراتی مانند زیر داشتیم:

```
c=a+b
if(a>=b) printf(a)
```

وقتی CPU می‌خواهد هرکدام از دستورات بالا را اجرا کند (بعد از هرکدام تبدیل به تعداد زیادی دستور سطح پایین شود که compiler این کار را انجام می‌دهد) یعنی وقتی CPU می‌خواهد c را با b جمع کند و در a بریزد ممکن است ما تا دستور زبان اسمبلی را بصورت ریاضی اجرا کنیم تا این اتفاق بیفتد. یا مثلاً CPU برای انجام عملیات مقایسه if مجبور شود ۲۰ تا ۳۰ دستور زبان اسمبلی اجرا کند.

زبان اسمبلی برخلاف زبان‌های سطح بالا برای CPU های مختلف متفاوت است چون ساختار داخلی هر CPU با هم متفاوت است. و در این جا ما فقط پردازنده 8086 (CPU) کار داریم. اولین قدم بررسی ثبات‌های داخلی CPU می‌باشد که شامل موضوعی قبل بود آن را تشکیل می‌دهند. پردازنده‌ی 8086 و 8088 حدوداً ۱۴ ثبات ۱۶ بیتی دارد برای داده‌ها نام AX و BX و CX و DX که این ۴ تا ثبات ۱۶ بیتی داده‌ای هستند. یعنی اطلاعات یا کدها در داخل آن‌ها ذخیره می‌شود یعنی مثلاً گفتیم می‌خواهیم دستوری را اجرا کنیم نمی‌توانیم به سطح بگوییم یکی را ضرب در ۵ جمع کن آخر قرار است ۵ بار جمع شود باید بگوییم:

یعنی عدد ۵ را داخل ثبات ax قرار بده `mov ax,5`

عدد ۷ را داخل ثبات bx قرار بده `move bx,7`

محتوای ثبات ax را با محتوای ثبات bx جمع کن `Add ax,bx`

جواب را در ax بریز

57

Scanned by CamScanner

همیشه دستوراتی که در زبان اسمبلی انجام می دهیم، عملوندهای ثبات، حافظه یا می باشد و یا حداقل یک عملوند باید ثبات باشد که در مثال بالا هر دو عملوند ثبات است. یکی ثبات ax و یکی ثبات bx است.
همانطور که گفتیم در یک CPU ثبات ها جزئی هستند که بسیاری از ثبات های CPU ثبات های داده ای هستند (data register) که این نوع ثبات ها برای عملوندها استفاده می شوند. مانند ax و bx و دستور جمع ما در این جا دو تا عملوند دارد که هر دو اطلاعات خام ما هستند. و اطلاعات خام همیشه باید داخل ثبات های داده ای ذخیره شوند. ما در این جا 5 را در یک ثبات داده ای و یا در ثبات داده ای دیگر قرار دادیم و بعد گفتیم این دو را با هم جمع کن. البته به صورت زیر هم می توانستیم بنویسیم

```
mov ax,5
add ax,7
```

حال این دو چه فرقی با هم دارند؟ در اولی هر دو عملوند ثبات هستند ولی در دومی یک عملوند ثبات و یک عملوند آنی یا immediate یا بلافاصله است به صورت زیر بنویسیم غلط است.

```
add 5,7
```

همانطور که گفتیم این پردازنده ۶ ثبات دارد که هر کدام ۱۶ بیتی می باشد. یعنی هر کدام از ثبات ها می توانند ۱۶ بیت حافظه داشته باشند یعنی بزرگترین عددی که در ثبات می توانیم ذخیره کنیم (در مبنای ۱۰) 32768 تا 32767 است که این دو تا ۱۶ بیت است یعنی ۱۶ تا یک.
۲- اگر نخواهیم علامت را لازم است باشد 65537 تا 0 که این ۱۶ تا یک می باشد.
در 8086 این امکان وجود دارد که از یک ثبات ۱۶ بیتی مثل AX هم استفاده کنیم تنها به این دارد که ۸ بیت کم ارزش را فراخوانیم یا ۸ بیت پر ارزش فقط در مورد ثبات AX اگر ۸ بیت کم ارزش را فراخوانیم می گوییم AL اگر ۸ بیت پر ارزش را فراخوانیم (سمت چپ) می گوییم AH کنترل آن را فراخوانیم می گوییم BX در مورد AX و CX و DX هم همین ترتیب
مثال: آن مثال زیر را داشته باشیم:
یعنی عدد 340 را در ۸ بیت کم ارزش ثبات BX ذخیره کن: mov bl, 340 ✗
البته دستور بالا غلط است چون 340 بیشتر از ۸ بیت حافظه دارد و ما ۸ بیت حداکثر عدد 255 را می توانیم ذخیره کنیم. مانند مثال زیر که صحیح است mov bl, 239 ✓

مثال زیر صحیح است:

```
mov bx,340 ✓
```

نوع دیگری از ثبات ها ثبات های آدرس هستند که در داخل آن ها عملوند ذخیره نمی شود بلکه آدرس عملوند ذخیره می شوند یعنی عملوند داخل محلی از حافظه است و ما از طریق آن ثبات می توانیم به آن محل دسترسی داشته باشیم. فعلاً فرض می کنیم که بخواهیم محتوای ثبات A را با خانه ی 145 حافظه جمع کنیم (نه با خود عدد) برای این کار ۲ راه داریم.

Scanned by CamScanner

۱-
Add a, [145]
برگردیم سر مثال اسمبلی بالا این معنا است که خود آی‌سی مالک نیست و همه‌ی ارجاعات به آی‌سی آدرس مستقیم است یعنی آی‌سی می‌رود خانه‌ی 145 حافظه را با ثبات a جمع می‌کند و جواب را در a بریزد. (همیشه ثبات اول مقصد است و حاصل جمع‌ها در آن ذخیره می‌شود)

۲-
ثبات آدرس
mov si, 145
Add a, [si]
در این جا در یک ثبات آدرس می‌سازیم. یعنی خود مقدار 145 را داخل si بریز و سپس a را با طرفی که آدرس به هم آن و si هم به 145 اشاره دارد، جمع کن.
در این جا داخل si آدرس عاوند ذخیره شده نه خود عاوند. پردازنده ۸۶ ۸۶ ۸۶ ثابت SP
DP - si - Di داریم که ۱۶ بیتی و از نوع address هستند یعنی در داخل این ثبات‌ها آدرس عاوند ذخیره می‌شود نه خود عاوند. البته آدرس‌های offset در این ثبات‌ها ذخیره می‌شود.
۶ ثبات دیگر با نام‌های ES - CS - SS - DS داریم که ۱۶ بیتی اند و ثبات‌های آدرس می‌باشند که در داخل آن‌ها آدرس‌های segment ذخیره می‌شود.
در اصل دو نوع آدرس داریم:
۱- offset address
۲- segment address
اگر خواهیم از آدرس‌ها offset استفاده کنیم از ثبات‌های SP - BP - si - Di و همچنین IP استفاده می‌کنیم، IP هم نوعی ثبات آدرسی است که آدرس offset دستور ذخیره می‌شود.
از طرفی اگر بخواهیم از آدرس‌های segment استفاده کنیم از ثبات‌های ES - CS - SS و DS استفاده می‌شود. در همه‌ی ما CPU واحد LAU یا واحد محاسبه و منطق داریم که کلیه کارهای عملیاتی و محاسباتی و منطقی را انجام می‌دهد. یعنی همیشه عاوند‌ها داخل ALU می‌آیند عمل خود را انجام می‌شود و جواب دوباره در ثبات مقصد قرار می‌گیرد.
و احد دیگری به نام واحد کنترل (control unit) داریم که ALU را جهت اجرای دادن عملکرد داخلی CPU برعهده‌ی control unit می‌باشد. عملیاتی مانند جمع و ضرب و ...

Scanned by CamScanner

جلسه‌ی دهم

زبانی که می‌خواهیم با یک پردازنده زبان اسمبلی و یا زبان ماشین صحبت کنیم اولین قدم این است که ثبات های اون پردازنده را شناسایی کنیم ببینیم چه ثبات هایی دارد و ثبات ها چند بیتی هستند و هر ثبات را برای چه کارهایی پیش بینی کرده و ساخته‌اند این ها از خیلی واقع الزامات کار است و دیدیم که ما هم CPU داریم و هر CPU زبان ماشین و اسمبلی مخصوص خودش را دارد. زبان ماشین ها تنوع زیاد است به تعداد CPU ها زبان ماشین مختلف داریم که طبیعتاً کار کردن و یاد گرفتن همه‌ی این ها امکان پذیر نیست. هنر ریز در معلومات پایه‌ای با مراجعه به راهنمای هر CPU بتوانیم زبان ماشین آن را بنویسیم و یا پردازنده‌ی آن را درک کنیم.

کلا در یک CPU ثبات های ما چند دسته‌اند در واقع به ۳ خانواده تقسیم می‌شوند ۳ اصل ثبات داریم.

۱- ثبات های داده‌ای هستند Data registers

۲- ثبات های آدرسی هستند Address registers

۳- ثبات پرچم flag registers

Data register: یکی از کلمه‌ها یا هر واحد اطلاعات در داخل این ثبات ذخیره می‌شود.

Address: خود عدد در داخل حافظه است که نشانه‌ی آدرس دارد. اول بیتی را وضعیت و آدرس در Address register ذخیره می‌شود.

ثبات پرچم (flag register) یک ثباتی است که تعدادی بیت دارد که ممکن است ۸ بیتی و ۱۶ بیتی باشد. بیت های آخر و هر کدام از این بیت های این ثبات معنی خاصی می دهد برخلاف دو تا ثبات دیگر این ثبات پرچم به صورت خودکار توسط CPU تغییر پیدا می کند هر CPU که برنامه‌ای اجرا می کند. بر اساس تعدادی اول دست چپ یا صفر یا یک بیت را تفسیر می دهد. هر واقع این ثبات را برای برنامه نویس بامزه و این ثبات ها می‌تواند مفید و منفعت آخرین دستوری که CPU اجرا کرده حتی کوتاه را پیش بینی کند.

هر بار که CPU دستوری را اجرا کند اتوماتیک bit های این ثبات تغییر پیدا می کند.

اصلا حالا همه bit های این ثبات flag یا پرچم می گویند به کل آن ها ثبات پرچم می گویند.

اینکه ما چند تا پرچم داریم در این ثبات بستگی به نوع ثبات و CPU دارد.

مثلا ما یک ثبات پرچم داشتیم در یک CPU فرضی یکی از پرچم های این ثبات bit ۷ است که به آن می گوییم zf و اگر zf = ۱ یعنی است و

اگر CPU دستوری را اجرا کند که حواب دستور صفر یا شد اتوماتیک مقدار ۱ قرار می گیرد و اگر CPU دستوری را اجرا کند که نتیجه‌ی آن غیر صفر یا شد اتوماتیک مقدار بیت صفر قرار می گیرد.

یکی از بیت های دیگر cf است. ثبات ها داخل CPU ها تعداد ۸ بیتی و ۱۶ بیتی یا شد.

مثلا در CPU ها ثبات های داده‌ای وابسته به نوع CPU ها می‌تواند AX و BX و CX و هم ۱۶ بیتی بعد. حالا می مثلا بیام دو تا عدد ۱۶ بیتی جمع کنم مثلا Add ax,bx مثل مثال، مثلا ax ۱۶ بیت و bx هم ۱۶ بیت است حالا وقتی دو تا ۱۶ بیتی را جمع می کنم ممکن جواب ۱۷ بیتی دربیاد یا ممکن دوهم بیاد و اگر دربیاد به اولین یک رقمی رقم نقلی یا گری می گویند.

		zf=0		CF	

cf = carry flag zf = zero flag

حالا اگر CPU دستوری اجرا کند که منجر به وجود آمدن رقم نقلی یا carry شود به طور اتوماتیک عدد در CF یک می‌گذارد. اگر اینطور نبود یعنی دستور یک رقم نقلی داشته و دیگر نه پس یعنی رقم نقلی نداشته
هر کدام از bit ها معنی خاصی می‌دهد اینکه چند تا هم پرچم داریم در CPU های مختلف می‌تواند متفاوت باشد
با توجه به شکل ۴ پرچم بیسیک شکل ۶ تا ثبات داده‌ای داریم AX و BX و CX و DX بنا ثبات‌های داده‌ای بودند. ۱۶ بیتی بودند به شکل ۸ بیت هم می‌توان استفاده کرد آیا ۸ بیت سمت راست ۸ بیت سمت چپ یا پسوند استفاده کرد.
اگر بگوییم AX یعنی کل ۱۶ بیت. اگر بگوییم AL یعنی ۸ بیت راست و اگر بگوییم AH یعنی ۸ بیت سمت چپ پرانرژی.
۶ تا ثبات دیگر داشت SP BP IP CS SI DI که ۱۶ بیتی بودند که ثبات آدرس بودند در داخل این قسمت آدرس offset نگهداری می‌شود
و ۶ تا ثبات دیگر داشتیم ES CS SS DS که ثبات‌های آدرسی هستند در داخل آن‌ها آدرس segment نگهداری می‌شود. حالا Address segment چیه و Address offset چیه؟
فرض: این ساختمان را در نظر بگیرید، طبقه دارد، ما به سه تا از طبقه‌ها می‌خواهیم بگوییم Address segment در داخل هر طبقه تعدادی کلاس داریم. ما به این کلاس‌ها می‌گوییم Address offset
کلاسی که داخلش هستیم segment طبقه چهار (segment 4) کلاس دوم (offset 2) و آدرس‌های offset می‌توانند تکراری باشند
در ۸۶ و ۸۶ خانه‌های حافظه را بخش بخش کرده که هر بخش segment می‌گوییم و در داخل هر بخش offset (شما به سادگی در نظر) وسیله offset ها می‌توانند تکراری باشند. سمت چپ segment و سمت راست offset در نظر
دارای پرچم و ثبات‌هایی SI DI BP SP IP آدرس‌های انتهای offset
می‌باشد و ثبات‌هایی ES SS CS DS آدرس‌های segment را در خود نگهداری می‌کنند.
همه CPU ها در حد ۸۶ و ۸۶ این واحدها حساب و منطق را دارند
ALU داخل CPU چی کار می‌کند؟ کلیه عملیات حسابی و منطقی را ALU حساب می‌کند مثل جمع و تفریق و ضرب و ... همگی توسط ALU دارند. CPU با ثبات‌های داده‌ای که وصل است مثلاً
AX و BX به داخل ALU و ALU عمل جمع را انجام بده و دوباره برگرداند داخل یکی از ثبات‌ها خیره کند.
واحد دیگر CPU یا (سیستم کنترل و امر اجرا) همان کنترل یونیت است.
CPU تایمی دهش فرمان یا دستوره کاری را انجام بدهد از control unit به ALU فرمان می‌دهد و control unit از همین برنامه نویس فرمان و دستور می‌گیرد.
ثبات پرچم به ALU وصل است. هر بار ALU عملی را انجام می‌دهد جواب را که می‌خواهد خیره کند بر اساس جواب ثبات پرچم هم تغییر می‌دهد.

Scanned by CamScanner

8086 ثبات پرچم (CPU) یک عدد داریم که 16 بیتی است ولی 9 بیت آن معنی دارد یعنی 9 تا flag داریم که داریم مطابق شکل زیر است

--	--	--	--	O	D	I	T	S	Z	--	A	--	P	--	C

ساختار پرچم در 8086

16 بیت جا داریم که 9 تای آن معنی دارد. عملگر Carry flag هست. S: اگر CPU دستوری که اجرا می کند جواب منفی باشد جواب مثبتی خارج می آید جواب منفی باشد اینجا یکی دارد. می برنامه نویس دستور اجرا شده را Sign flag نگاه می کند براساس صفر و یک بودن اول جواب آن مثبت یا منفی بودن را می آورد.

D در 8086 ما دستوراتی داریم برای مکرر کار با رشته، هر رشته مجموعه ای از کاراکتر است. مثلاً دستور بدهیم برود داخل یک رشته دنبال یک کاراکتر خاص بگرد. وقتی ما خواستیم یک رشته را پردازش کنیم می گوییم دوتا آدرس داریم رشته را از راست به چپ پردازش کنیم یا از چپ به راست. مثلاً اگر داریم رشته (بیت رفتن است) دنبال عزت (ت آخرش) اگر از راست به چپ بگردیم اگر از راست به چپ بگردیم (ت) را پیدا می کند اگر از چپ به راست بگردیم (ت) است را انتخاب می کند.

اگر D=0 باشد رشته ای را از چپ به راست جستجو می شود اگر D=1 باشد رشته را از راست به چپ.

O (سرریز) آخرین دستور که اجرا کردیم حالت سرریز داشته باشد این بیت 1 است وگرنه صفر است.

ثبت دستورالعمل (Instruction queue)

در 8086 آمده یک جور عملیات خط لوله انجام داده است. عملیات خط لوله یعنی دریک لحظه چند کار را انجام میدهد. در یک CPU ما برای اجرای دستور سه مرحله داریم. اول دستور را از حافظه می آوریم داخل CPU بعد Decode می کردیم بعد دستوری است بعد اجرایی کردیم. در 8086 آمده در صف دستور عمل است اومده تا بیت گرفته نشده که دستور اول را Fetch می کند و دستور اول میرود در مرحله Decode دستور دوم را Fetch می کند می آید در این حافظه دستور اول میرود به مرحله ای که دستور دوم میره در مرحله Decode و دستور سوم میره در مرحله Fetch

Fetch	Decode	Execute			دستور اول
	Fetch	Decode	Execute		دستور دوم
		Fetch	Decode	Execute	دستور سوم

واکشی دستور اول
واکشی دستور دوم / کشف دستور اول
واکشی دستور سوم / کشف دستور دوم / اجرای دستور اول

Scanned by CamScanner

این باعث می‌شود دستورات اجرای دستورات بالابرد و ما به این عمل خط لوله می‌گوییم
در همه‌ی CPU انواع اقسام خط لوله‌ها را داریم و یکی از تفاوت‌های 8086 و 8088 در همین
صف دستورالعمل است. در 8086 این صف دستورعمل 6 بایت است در 8088 صف دستور
عمل 4 بایت است.

ثبات AX (Accumulator) ثبات انباره‌ای است. معمولاً ثبات انباره یکی از ثبات‌های داده‌ای
است که جواب داخلش ذخیره می‌شود و حتی از دستورالعمل‌های ورودی و خروجی و محاسباتی از این ثبات
استفاده می‌کنند.

ثبات BX (Base) ثبات پایه است که علی رغم این که این ثبات داده‌ای است ولی
در پیاده‌سازی آدرس دهی هم از آن استفاده می‌شود به عنوان آدرس پایه

ثبات CX (counter) از این ثبات در حلقه‌هایی که خواهیم شمارش انجام بدهیم به عنوان شمارنده
حتماً باید از آن استفاده کنیم

ثبات DX (Data) به عنوان ثبات داده‌ای از آن استفاده می‌شود.

ثبات پرچم: هر CPU یک ثبات پرچم دارد که وضعیت پردازنده بعد از انجام عمل را نشان می‌دهد.

ثبات C (carry) اگر جواب حاصل داشته باشد بیشتر از دو تا 16 بیتی یا 8 بیتی کاری نیست
مثلاً اگر ما بگوییم add AL,BL حاصل آن جواب 9 بیت نگه می‌دارد carry حاصل آن add AX,BX جواب
8 بیت 16 بیت

1 است شود carry می‌شود بیشتر از دو تا حاصل 8 بیتی است یا 16 بیتی

ثبات P: براساس تعداد یک‌های جواب این بیت به گونه‌ای صفر یا یک می‌شود که تعداد یک‌های موجود در
جواب به همراه این بیت P زوج باشد (Parity)

10110001 تعداد یک‌ها زوج

پس P=1

بیت A (Auxilary carry): بعضی از CPU ها carry bit 4 را در پیاده‌سازی خود
برای عملیات در داده‌های 8bit به این A می‌گویند

فرض کنید می‌خواهیم:

```
  1 0 0 0 1 0 1 0
+ 0 0 1 1 1 0 0 0
-----------------
  1 0 0 0 0 0 1 1
```

یعنی 4bit اول با 4bit دوم جمع شد Auxilary داشت.
کدهای BCD 4 بیتی بودند اگر carry داشتیم AF=1 و اگر نداشت AF=0

بیت S (sign) اگر جواب مثبت باشد S=0 و اگر جواب منفی باشد S=1 است

بیت I (Interrupt): بعضی از وقفه‌ها maskable اند یعنی می‌توانیم از برنامه‌ی اصلی
با صفر کردن این بیت I

کامپایلر فرنوس 5 یک برای اولین وقفه‌های maskable از کار می‌افتد اگر I یک وقفه‌ها
به طور عادی به کار خود ادامه می‌دهند.

Scanned by CamScanner

I non maskble کاری ندارد

* بیت O (over flow) بیت سرریز است. حالتی است که حاصل یک عدد بزرگ است که بیت علامت خراب می‌شود.

مثلاً O شیفت می‌خورد یا مکمل می‌شود صفر

01110000 + 01000001 → 10110001

دو عدد ۸ بیتی با هم جمع شدند و حاصل ۸ بیت شد پس carry نداریم ولی علامت تغییر کرد پس over flow داریم.

مثبت + مثبت = منفی

IP یا Instruction Pointer در واقع ایفا کننده آدرس offset دستور عملی که قرار است اجرا شود را ذخیره می‌کند.

segment offset

حافظه را زنگ بزنید. دیسیم در 8086 ما ۲۰ تا خط Address داریم پس می‌توان نتیجه گیری کرد همه‌ی خونه‌ها از صفر تا $2^{20}-1$ فرض کنید یک segment داشتیم

0			
1			
2			
...			
....			
...			
...		0	→ آدرس offset
3200		1	
...		2	3200 → خونه‌شماره‌ی 2 از segment 3200
...		3	
3232		...	
		...	
		...	
		$X_{MAX} = 64 * 1024 = 2^{16} - 1 = ffff$	
$64K$ $2^{20}-1$			
FFFFF			

Scanned by CamScanner

قسمت زنده شده segment است

در 8086 همیشه آدرس شروع segment برعدد ۱۶ بخش پذیر باشد یعنی هر جایی نمی‌توانیم شروع segment را در نظر بگیریم

تعداد خانه‌های segment یا همون offset ها میشه 2^{16} تایی باشد.

$X_{max} = FFFF * 1024 = 2^{16} - 1$

یعنی بزرگترین آدرس offset $2^{16} - 1$ است. صفر جزو اعداد حساب میشه

هر segment از طریق 3200 segment آدرس داد هم 3232 segment

یک segment هر جای حافظه می‌تواند باشد

هیچ segment، offset، segment نمی‌شود شماره‌ی خانه‌ی حافظه را

آدرس‌های segment، offset است

بزرگترین آدرس offset ($2^{16} - 1$) است در مبنای هگز می‌شود FFFF که همون 65535 یک است که می‌شود ۱۶ تا یک

آدرس offset خانه‌ی ۱۶ بیتی است بنابراین در 8086 ثبات‌های SP BP SI DI IP که همه آدرس‌های offset توضیح داده شد ۱۶ بیتی گرفتیم

آدرس segment بزرگترین آدرسی که در 8086 داریم $2^{20} - 1$ است در مبنای ۱۶ می‌شود FFFFF

که می‌شود ۲۰ bit و آدرس segment که هر عددی می‌تواند باشد

آدرس segment یک عدد ۲۰ bit است که بر ۱۶ بخش پذیر است.

با توجه به اینکه آدرس segment ها ۲۰ بیتی است پس در ثبات‌های ES CS SS DS را که آدرس segment ها در آن ذخیره می‌شود را ۱۶ bit گرفتند چرا ۲۰ بیتی نگرفتند

این بحثی که کردیم به طبیعت اعدادی که بر ۱۶ بخش پذیر است

که ۱۶ را به مبنای هگز ببریم: $16 = 10_H$ $32 = 20_H$

همیشه عددهای بر ۱۶ بخش پذیر هستند در مبنای ۲ ۴ تا صفر در سمت راست دارند در مبنای ۱۶ یک صفر در سمت راست دارند.

آدرس segment چه خاصیتی دارند؟ ما می‌گوییم یک عدد ۲۰ bit است که ۴ تا صفر سمت راست دارند.

XXXXXXXX XXXXXXXX 0000 = FFFF0H

هر عدد مبنای ۲ این که بر ۱۶ بخش پذیر باشد حتماً ۴ تا صفر سمت راست دارد

ما در مبنای هگز نمی‌خواهیم بگیریم

آدرس segment یک عدد ۲۰ bit است که حتماً ۴ بیت سمت راست صفر است و چون ما می‌خواهیم دیگر صفرها را ذخیره نمی‌کنیم به همین علت ثبات‌های segment را ۱۶ بیتی گرفتند.

Scanned by CamScanner

بنابراین مقدار خواهیم آدرس را محاسبه کنیم مقدار اول ثبات سگمنت را در ۱۶ ضرب و سپس به آفست اضافه می‌شود تا آدرس محاسبه شود.

مثال: آدرس بعد از حرکت ۲۵ خانه جلو از A32B:4DF5 را بدست آورید.

A32B0 +
4DF5

A80A5 +
25

A80CA

5 = 5 + 0

F + B = ۲۷ بعدی ۱۶ باقی ۱۰ اضافه میشه A

۱۶ = D + 2 + 1

A = 13 = 4 + 3 + 1

غلط است چون A80A5 مبنای ۱۶ است که ۲۷ مبنای دهدی میشود باید جمع زد باید ۲۵ را به مبنای ۱۶ برد.

25 = 19H

A80A5 + 19 = A80BE ✓

Scanned by CamScanner

جلسه‌ی یازدهم (۹۵/۹/۱۸)

- چون در ثبات‌های آدرس، آدرس‌های segment، حجیم‌اند ولی ۱۶ بیت در آن‌ها ذخیره می‌شود و چون ۴ بیت سمت راست آن‌ها صفر می‌پذیرند ۱۶ بخش‌پذیرند.
- آدرس‌های offset همواره ۱۶ بیت‌اند چون 64K یا در اصل همان ۱۶ تا یک هشتند.
- آدرس‌ها در ثبات‌ها به جای ۲۰ بیت، ۱۶ بیت‌اند چون حافظه‌ی کنونی می‌شناخت یعنی 64K یا MB1 هرچقدر address bus بزرگتر باشد، فضا یا تراکم آدرس‌دهی آن بیشتر است.

* در سیستم‌های مبتنی بر پردازنده‌ی 8086 هر برنامه برای این‌که اجرا شود، در هر لحظه از ۴ تا segment می‌تواند استفاده کند. segment ها نواحی از حافظه هستند که آدرس شروع آن‌ها بر ۱۶ بخش‌پذیر است و اندازه‌ی آن‌ها حداکثر 64K می‌باشد.

حال می‌خواهیم این ۴ تا segment را بررسی نماییم:

۱) code segment ← داخل این segment خود برنامه ذخیره می‌شود یا به عبارتی دیگر دستوراتی که در یک برنامه می‌خواهد اجرا شود در داخل این قسمت ذخیره می‌شود و CPU همیشه از code segment فرمان می‌گیرد.

۲) data segment ← اطلاعاتی که قرار است پردازش شود و همچنین اطلاعاتی که پردازش شده (جواب - نتایج) در داخل data segment ذخیره می‌شود یعنی داده‌های خام + جواب.

۳) stack segment ← معمولاً برنامه‌ها برای اجرا به stack یا پشته هم نیاز دارند.

(stack یا پشته نوعی حافظه است؛ همه‌ی حافظه‌ها از نوعی هستند که باشند برای دستیابی به اطلاعات‌شان به آدرس‌ها نیاز دارند. حافظه‌هایی که ما در این جلسه می‌خواهیم بررسی کنیم Random Access Memory یا RAM است یعنی دسترسی به حافظه‌ها به صورت تصادفی است، فقط به شرطی که در محدوده‌ی آدرس‌دهی حافظه قرار بگیرند.)

حالا نوعی دیگر از حافظه به نام حافظه‌ی stack وجود دارد که تا قبل دسترسی در آن به این شکل می‌باشد که ما در هر لحظه فقط به یک خانه دسترسی داریم، یعنی حق داریم از اطلاعات آن خانه استفاده کنیم و یا بعد از آن خانه یا روی آن خانه اطلاعاتی را بنویسیم، مثلاً اگر یک stack داشته باشیم نمی‌توانیم یکسره خانه‌ی ۴۰ام یا ۵۰ام را فراخوانی کنیم بلکه باید به این ترتیب عمل کنیم که آخرین اطلاعاتی که در آن ذخیره شده است و ما فقط به آن جا دسترسی داریم. همچنین stack این‌گونه است، به همین خاطر که در آخر ذخیره می‌کنیم می‌توانیم دسترسی داشته باشیم و می‌توانیم آن را برداریم یا روی آن اطلاعاتی را قرار دهیم. در واقع stack را می‌توان به یک سری بشقاب تشبیه کرد که روی هم ذخیره شده‌اند و وقتی بشقابی را می‌خواهیم برداریم فقط اولی را که در رأس قرار دارد می‌توانیم برداریم و یا روی آن بشقاب آخری می‌توانیم بشقاب‌های دیگری را قرار دهیم که اصطلاحاً در stack به عمل ذخیره کردن اطلاعات عمل push و به عمل بازیابی از stack عمل pop می‌گویند. مادر مثال بشقاب، حاضر نیستیم بشقابی را میان بشقاب‌های دیگر قرار دهیم و یا از وسط آن‌ها نمی‌توانیم یک بشقاب بیرون بکشیم و در stack هم همین شکل می‌باشد.

Scanned by CamScanner

همه‌ی برنامه‌ها همانطور که گفتیم حتما باید یک segment به نام stack داشته باشند. در 8086 یک segment به نام stack پیش‌بینی شده است یعنی یک فضایی از حافظه‌ی اکثر به طول 64K به عنوان stack برنامه‌ی ما عمل می‌کند.

Extra segment ← ممکن است علاوه بر segment های موقتی برنامه‌ی ما به یک سری segment اضافی هم نیاز داشته باشد که segment اضافی نام دارد. یعنی مثلا ممکن است حجم داده‌هایی که قرار است پردازش شود زیاد باشد و در داخل یک segment نشود. البته استفاده از این نوع segment الزامی نیست و بستگی به نوع برنامه و دستوراتی که در داخل آن می‌باشد دارد. چون برخی از دستورات فقط با Extra segment کار می‌کنند و یا data segment کار می‌کند پس باتوجه به نوع دستورات و این که آیا برنامه‌ی ما به Extra segment نیاز دارد یا خیر ممکن است برنامه‌ی ما Extra segment داشته باشد و برعکس.

پس همه‌ی برنامه‌های در حال اجرا از این 3 segment استفاده می‌کنند و از هم جدا هستند یعنی آیا مثلا در CPU 3 برنامه در حافظه ذخیره شده 3 تا code segment، 2 تا data segment و 2 تا extra segment داریم.

- در ثبات‌های آدرسی، آدرس code segment برنامه‌های در حال اجرا در ثبات CS ذخیره می‌شود.
- آدرس data segment برنامه‌های در حال اجرا در ثبات DS ذخیره می‌شود.
- آدرس stack segment برنامه‌های در حال اجرا در ثبات SS ذخیره می‌شود.
- آدرس برنامه‌ای که extra segment استفاده کند آدرس extra segment برنامه‌ی در حال اجرا در ثبات ES ذخیره می‌شود.

*** آدرس هافظه به برنامه‌ها چگونه switch کند. تعاریف تمام این‌ها باید عوض شود.

- وقتی می‌گوییم: "آدرس offset" offset به تنهایی معنی نمی‌دهد بلکه باید مشخص کنیم کدام segment صددرصد است.

آدرس مشغول از 2 بخش می‌باشد: segment : offset

یعنی خانه‌ی صددرصد در داخل کدام segment است و در کل segment چند offset ای وجود دارد یا چنین خانه‌ای است.

ثبات SP همیشه می‌گوید خانه‌ی Top of stack کجاست یعنی آن خانه‌ای که آخرین بار اطلاعات در آن ذخیره کردیم کجاست پس Stack Pointer داخل segment ای به نام stack segment کار کند یعنی به معنی در داخل stack segment اشاره می‌کند چون آدرس Top of stack را به صورت زوج می‌دهد یعنی SS و SP هستند هم هستند چون SP به معنی در داخل Top of stack اشاره می‌کند

Scanned by CamScanner

- در ثبات IP یا Instruction Pointer آدرس دستورالعملی قرار است اجرا شود در حال اجرا.
- دستورات در Code Segment قرار دارند و ثبات IP همیشه در داخل Segment ای آدرس دهی می کند که CS به آن اشاره می کند. CS و IP همیشه با هم هستند
- ثبات BP یا Base Pointer معمولا در داخل stack segment اشاره می کند و همیشه به stack segment اشاره می کند. Base Pointer یک ثبات است که در عملیات آدرس دهی و آدرس دهی غیر مستقیم از آن به عنوان آدرس استفاده می کند.
- SI یا Source Index معمولا در داخل حافظه اشاره می کند همیشه با Data Segment متناظر است.
- DI یا Destination Index همیشه معمولا در داخل Extra segment اشاره می کند و segment پیش فرض آن Extra segment می باشد.

** مادر این جلسه می خواهیم آرگومان دستورات segment های پیش فرض را تغییر دهیم.

در اسمبلی 8086 دستورات به چند بخش تقسیم می شوند

1. گروه دستورات انتقالی
2. دستورات محاسباتی
3. دستورات کنترلی (پرش)
4. دستورات .....

دستورات انتقالی:

متداول ترین دستوری که در دستورات انتقالی داریم دستور move می باشد. این دستور برخلاف اسمش عمل copy را انجام می دهد و در اصل یک تابع است که به آن پارامتر می دهیم destination مقصد و source یا مبدأ.

یعنی در این دستور همیشه اول مقصد و source باشد در مقصد copy می شود این مبدأ و مقصد نسبت به هم یک رابطه داشته باشند.

1) هر دو باید هم اندازه باشند یعنی یا باید به اندازه ی 1 byte باشند یا به اندازه ی 1 word باشند. به مثال زیر توجه کنید:

mov dest, source

*mov ax, bx

* یعنی محتوای ثبات bx را در داخل ثبات ax کپی کن. در این جا ax مبدأ و bx مقصد می باشد و هر دو 16bit هستند

**Mov al, ch

با هم داریم:

** در این حا هم دو هم اندازه و 8bit می باشند

***Mov dx, ah

*** حالت بالا قبول نیست چون هم اندازه نیستند

Scanned by CamScanner

۲- مبدأ و مقصد چه حالت‌هایی تعاریف را داشته باشند

الف) حالت Register to Register ← RR مبدأ و مقصد ثبات باشند
مثال‌های ما: mov ax,bx / mov al,ch همه دو ثبات‌اند.

ب) Register Immediate ← RI یکی ثبات و دیگری یک مقدار معلوم باشد مثال:
mov bh,500
*** باید توجه داشت که مقدار معلوم باید به گونه‌ای باشد که در داخل ثبات مقصد جا شود. مثلاً نمی‌توان گفت
mov bh,500 چون 500 از 8 بیت بیشتر است.

ج) Register to memory ← RM مبدأ ثبات و مقصد memory می‌باشد
مثلاً داریم:
mov al,[300]
خط بالا می‌گوید: محتوای خانه‌ای که آدرس آن 300 است را در داخل Data segment را در ثبات al بریز.
اگر بخواهیم segment پیش‌فرض را عوض کنیم به مثال زیر دقت کنید:
mov al,es:[300]
یعنی محتوای خانه‌ی 300 در داخل Extra segment را در al بریز.
در واقع در حالت عادی چون تکلیف کار خانه‌ی Data segment است segment را به عنوان پیش‌فرض می‌گیرد و دیگر در دستور حالت segment پیش‌فرض را Extra segment فرض نمی‌کنیم.
گاهی اوقات هم می‌توانیم از روش آدرس‌دهی غیرمستقیم استفاده کنیم. مانند مثال زیر:
mov di,300
mov al,[di]
Extra segment حول di حول Extra segment داخل 300 در segment آدرس‌دهی می‌کند. به جای di می‌گفتیم si آن وقت پیش‌فرض Data segment بود.
مگر این که segment پیش‌فرض را عوض کنیم و بگوییم:
mov di,300
mov al,ds:[di]
یعنی محتوای خانه‌ی 300 در داخل Data segment را در ثبات al بریز

د) memory to Register ← MR مبدأ memory و مقصد Register باشد.
mov [300],al
محتوای ثبات al را بریز در خانه‌ی شماره‌ی 300 در داخل Data segment چون segment را مشخص نکردیم.
mov si,4000
mov [si],ch

Scanned by CamScanner

ثبات Ch بمقدار 400 در داخل Data segment می‌رود چون از si استفاده کردیم

```
mov si,400
mov es:si,ch
```

معنای ثبات Ch را بریز در خانه‌ی شماره‌ی 400 در داخل Etra segment چون es را آدرس segment پیش‌فرض را عوض کردیم و ما آدرس es گذاشتیم

memory یا MI یا memory Immediate (ج

```
mov [700],24
```

عدد 24 را در خانه‌ی 700 Data segment ذخیره کن

```
mov bp,250
```

عدد 12 را در خانه‌ی شماره‌ی [250] در داخل stack segment ذخیره کن چون پیش‌فرض bp و ss می‌باشد.

```
mov bp,12
mov ds:bp,12
```

12 را در داخل خانه شماره 250 در داخل Data segment ذخیره کن چون در اینجا دستور پیش‌فرض را تغییر دادیم

Scanned by CamScanner