# Negar - Lecture 16

## سوال اول

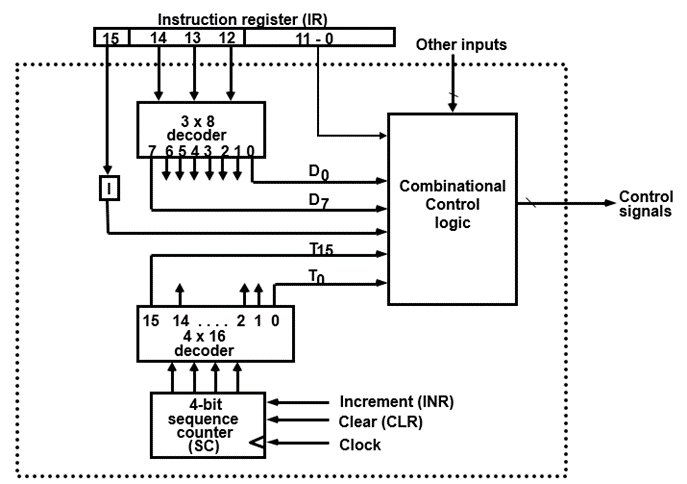
در یک کامپیوتر Control Unit چه وظیفه ­ای دارد؟ پیاده­ سازی این قسمت به طور معمول به دو روش می ­تواند انجام شود؛ این دو روش را به اختصار توضیح دهید.

می دانیم هر دستور پردازنده باید به تعدادی دستور سطح RTL شکسته شود. در یک کامپیوتر Control Unit یا CU وظیفۀ ترجمۀ دستورات زبان ماشین به سیگنال های کنترلی برای ریزعملیات هایی که آن دستور را اجرا می کنند دارد.

Control Unit ها به طور معمول به دو روش پیاده سازی می شوند:

1. حالت Hardwired: در این حالت یک مدار ترکیبی/ترتیبی طراحی شده و این مدار همان پیاده سازی واحد کنترلی می باشد. اجزاء و نحوۀ پیاده سازی این مدار پس از ساخته شدن نمی تواند تغییری بکند.
2. حالت Microprogrammed: در این حالت یک حافظه درون پردازنده داریم که قابل program کردن است و می توان logic آن را تغییر داد (چیزی شبیه به منطق FPGA)، بنابراین می توان با program کردن این حافظه ریزعملیات ها را تعریف کرد. در کنار این حافظه یک مدار ترکیبی وجود دارد که با خواندن هر خانه از حافظه یک ریزعملیات را اجرا می کند.

**سوال دوم(+)**

با توجه به شکل داده شده به سوالات زیر پاسخ دهید.

الف) decoder 3x8 موجود در قسمت بالای شکل چه وظیفه ­ای دارد؟

ب) کدام قسمت از مدار گام­ های دستورات را مشخص کرده و به بخش Combinational Control Logic حالت ترتیبی می ­دهد؟ کارکرد این بخش را توضیح دهید.

ج) فرض کنید طولانی ترین دستوری که برای پردازنده تعریف شده است 7 کلاک طول می ­کشد. در این صورت Sequence Counter حداقل باید چند بیت داشته باشد؟

د) مقدار موجود در Sequence Counter در چه زمان ­هایی باید Clear شود؟

الف) با توجه به شکل بالا بیت های 12، 13 و 14 از IR مربوط به Opcode دستوری هستند که باید اجرا شود. این 3 بیت به یک دیکدر 3 به 8 می روند و خروجی این دیکدر مشخص می کند که کدام دستور باید اجرا شود. برای مثال اگر سیگنال D3 برابر با یک بوده و باقی سیگنال های D0 تا D7 برابر با صفر باشند دستور شماره 3 باید اجرا شود. در ادامه هر کدام از این 8 سیگنال خروجی بخشی از مدار سمت راست را enable می کنند که به دستوری که باید اجرا شود مربوط است، بنابراین با یک شدن D3 در این مثال تنها بخشی از مدار ترکیبی سمت راست فعال می شود که برای اجرای دستور سوم لازم است.

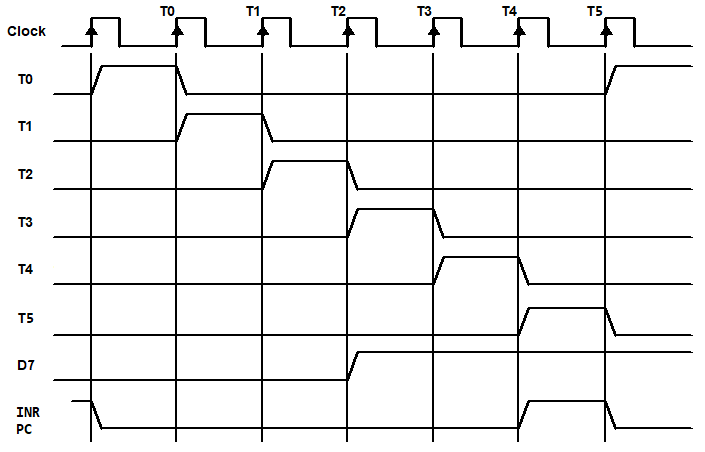
ب) می دانیم هر دستور در قالب یک گام نمی تواند اجرا شود و باید مراحلی را طی کند. لذا مدار ترکیبی واحد کنترل باید بداند در چه کلاکی قرار دارد تا در کلاک های متفاوت بخش های متفاوتی از این مدار enable شوند. قسمت پایین مدار (SC و decoder 4x16 در این شکل) این گام ها را تعیین کرده و به مدار حالت ترتیبی می دهد. در حقیقت این که واحد کنترل بداند در چه مرحله ای است را این بخش مشخص می کند. SC یک شمارنده است که از صفر شروع به کار کرده و به ترتیب می شمارد. خروجی این شمارنده به ورودی یک دیکدر وارد شده و در نتیجه 16 سیگنال از دیکدر خارج می شود که در هر لحظه از زمان تنها یکی از آن ها فعال است. یک بودن هر یک از سیگنال های T0 تا T15 مشخص می کند در کدام گام از دستور مورد نظر قرار داریم. SC به ازای هر دستور از 0 تا تعداد مراحل دستور (که برای دستورات متفاوت می تواند متفاوت باشد) می شمارد و پیش از اجرای هر دستور دوباره مقدار صفر به خود می گیرد تا هر دستور بتواند از گام اول خود شروع به اجرا کند.

ج) همانطور که در قسمت قبل نیز اشاره شد SC باید بتواند تعداد مراحل هر دستور بشمارد. از آن جایی که طولانی ترین دستور پردازنده 7 کلاک طول می کشد (T0 تا T6) شمارنده باید قابلیت شمردن از عدد 0 تا 6 را داشته باشد و لذا باید حداقل 3 بیتی باشد.

د) همانطور که در قسمت "ب" اشاره شد SC برای مدار ترکیبی واحد کنترل مشخص می کند که در کدام گام از دستور قرار دارد. بنابراین پیش از اجرای هر دستور جدید مقدار موجود در SC باید برابر با صفر قرار گیرد تا دستور از گام اول خود شروع به اجرا کند (این کار به طور معمول در انتهای اجرای هر یک از دستورات صورت می گیرد تا SC برای دستور بعدی آماده شود).

**سوال سوم(امتحان)**

ریزعملیاتی که آن چه در شکل زیر اتفاق می ­افتد را توصیف می ­کند چیست؟ ابتدا مراحل چهارگانۀ مربوط به اجرای یک دستور در کامپیوتر پایه را شرح داده و سپس مشخص کنید به چه علت سیگنال D7 پس از کلاک دوم (و نه از همان ابتدا) در اختیار ما قرار گرفته است.

\* راهنمایی: با یک شدن مقدار پایۀ INR رجیستر PC، مقدار آن یک عدد افزایش می ­یابد.

ریزعملیات مربوط به شکل:

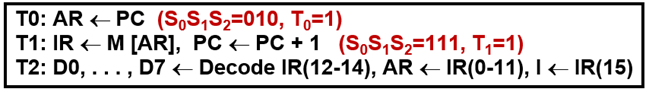
D7.T5: PC ← PC + 1

به طور کلی اجرا هر دستور چهار مرحلۀ کلی دارد:

1. مرحلۀ Fetch کردن دستور از حافظه: در این مرحله دستور از حافظه درون IR لود می شود.
2. مرحلۀ Decode کردن دستور: در این مرحله Opcode مربوط به دستور Decode شده و تشخیص می دهیم دستور موجود در IR چه دستوری است.
3. خواندن Operand ها: در این مرحله عملوندهای دستور (در صورت نیاز از حافظه) خوانده می شوند.
4. اجرای دستور: در این مرحله دستور با توجه به اطلاعات بدست آمده از مراحل قبل اجرا می شود.

یک شدن سیگنال D7 به این معناست که دستور هفتم پردازنده باید اجرا شود و همانطور که اشاره شد این که کدام دستور باید اجرا شود بسته به Opcode دستور موجود در IR دارد. بنابراین یک شدند D7 پس از Decode کردن دستور یعنی مرحلۀ دوم اتفاق خواهد افتاد و پیش از آن نمی دانیم دستوری که باید اجرا شود چیست و کدام یک از سیگنال های D0 تا Dn یک خواهند شد.

**سوال چهارم(-)**

با توجه به مراحل Fetch و Decode کردن یک دستور در کامپیوتر پایه، که در ادامه آمده ­است، به سوالات زیر پاسخ دهید.

الف) به چه دلیل زمان Fetch کردن یک دستور باید ابتدا مقدار PC را درون رجیستر AR قرار دهیم و به طور مستقیم از PC استفاده نمی ­کنیم؟

ب) همانطور که مشاهده می ­کنید یکی از ریزعملیات­ هایی که باید در این قسمت اجرا شود ریختن 12 بیت کم ارزش رجیستر IR در رجیستر AR است. ابتدا توضیح دهید این کار به چه منظور انجام می­شود و سپس تحلیل کنید که آیا وجود این ریزعملیات برای تمام دستورات الزامی است یا خیر.

ج) آیا ممکن است در روند اجرای یک دستور پس از زمان T1 رجیستر IR بار دیگر مقداردهی شود؟ توضیح دهید.

الف) در زمان Fetch کردن دستور می خواهیم آن را از حافظه بخوانیم، بنابراین باید آدرس دستوری که قرار است اجرا شود (و تا به اینجا در PC قرار دارد) را به حافظه اعلام کنیم. رجیستر PC به حافظه متصل نیست و نمی تواند مقدار موجود در خود را به عنوان آدرس به حافظه اعلام کند، اما در مقابل رجیستر AR به حافظه متصل است، بنابراین در ابتدا مقدار PC را به AR منتقل می کنیم و سپس مقدار خانۀ مربوطه از حافظه (که همان دستور مورد نظر ماست) را از این طریق می خوانیم.

ب) می دانیم در بعضی از دستورات پردازنده 12 بیت کم ارزش دستور آدرس Operand آن در حافظه است. حال در این مرحله 12 بیت کم ارزش دستور را داخل AR می ریزیم تا در ادامه در صورت نیاز بتوانیم مقدار آن را از حافظه بخوانیم. نتیجۀ این کار تنها زمانی برای ما مفید است که در حال اجرای چنین دستوری باشیم که 12 بیت پایین آن واقعا آدرس خانه ای از حافظه است، اما ما فارغ از نوع دستور از فرصت استفاده کرده و این کار را انجام می دهیم (زیرا در این کلاک گذرگاه آزاد است و به مقدار درون AR نیازی نداریم و می توانیم آن را بازنویسی کنیم). پس از آن اگر نیاز بود از این مقدار موجود در AR استفاده می کنیم و اگر نه ضرر نمی کنیم. در مقابل اگر می خواستیم بررسی کنیم که این کار نیاز است یا خیر و سپس این 12 بیت را داخل AR کپی کنیم تعداد گام های اجرای دستور افزایش می یافت.

ج) خیر. در طول اجرای یک دستور یک و فقط یکبار چیزی که از حافظه می خوانیم را دستور تعبیر می کنیم و پس از آن هر آن چه از حافظه خوانده شود یک Operand است. بنابراین رجیستر IR که وظیفه دارد دستورات را در خود نگهداری کند فقط در زمان T1 مقداردهی خواهد شد.

**سوال پنجم(امتحان)**

دستور Branch and Save Return Address یا همان BSA و دستورات متناظر با آن در دیگر کامپیوترها یکی از دستورات مهم و پرکاربرد هستند. کاربرد اصلی این دستور برای انجام چه کاریست؟ با فرض این که برای انجام این دستور از ریزعملیات زیر استفاده می ­شود روندی که این دستور طی می ­کند را به طور کامل توضیح دهید.

**BSA:**

D5T4: M[AR] ← PC, AR ← AR + 1

D5T5: PC ← AR, SC ← 0

از این دستور و ترکیب آن با دستور BUN برای فراخوانی توابع (یا به طور دقیق تر subroutine ها) می توان استفاده کرد. زمان فراخوانی یک تابع می خواهیم به قسمت دیگری از کد پرش کرده و در انتها با دستور return به خطی از کد که پیش از آن در حال اجرای آن بودیم بازگردیم. این دستور می تواند چنین امکانی را برای ما فراهم کند.

با توجه به ریزعملیات داده شده میتوان متوجه شد که در این کامپیوتر به طور قراردادی برای تعریف یک subroutine در خط اول آن آدرس خطی از کد که باید پس از اجرای تابع به آن برگردیم (return address) و در خطوط بعدی بدنۀ subroutine قرار خواهند گرفت.

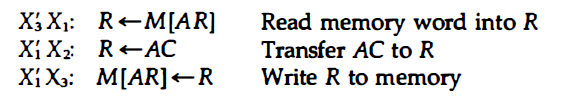
در زمان T4 مقدار PC را درون جایی ذخیره می کنیم که مقدار درون رجیستر AR (یا همان operand ورودی دستور) به آن اشاره می کند. سپس مقدار AR را یکی افزایش داده و در زمان T5 آن را درون PC ذخیره می کنیم. با این کار دستور بعدی ای که اجرا خواهد شد در مکانی درست پس از مکانی قرار دارد که return address را درون آن ذخیره کرده ایم. در نهایت نیز مانند هر دستور دیگر مقدار SC را صفر می کنیم.

برای تکمیل این فرآیند در انتهای کد مربوط به subroutine یک BUN غیر مستقیم قرار می دهیم که به خانۀ اول subroutine (همان خانه ای که return address را درون آن ذخیره کرده بودیم) اشاره می کند. با این کار پس از اتمام تابع آدرس دستوری که پیش از فراخوانی تابع در حال اجرای آن بودیم را بدست آورده و به آن پرش می کنیم تا به ادامۀ دستورات بپردازیم.

# Mahvash - Lecture 18

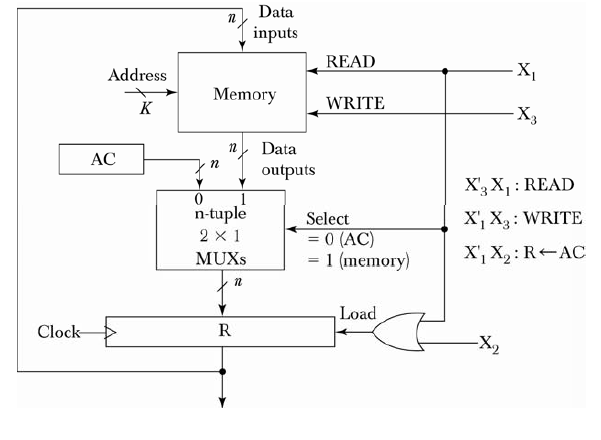
## سوال اول(ـ)

دستورات register transfer در ارتباط با ثبات R و حافظه به شرح زیر است.



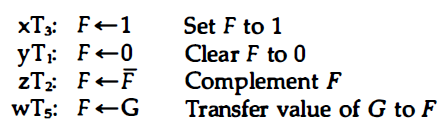
حافطه شامل ورودی داده، خروجی داده، ورودی آدرس و ورودی سیگنال کنترلی برای خواندن و نوشتن است. پیاده سازی سخت افزاری ثبات R و حافطه را به شکل block diagram را رسم کنید. نشان دهید چگونه توابع کنترلی x1 تا x3 کنترل load ورودی ثبات R، خطوط select در مالتی پلکسر و ورودی های خواندن و نوشتن در حافظه را کنترل میکنند.

پاسخ:



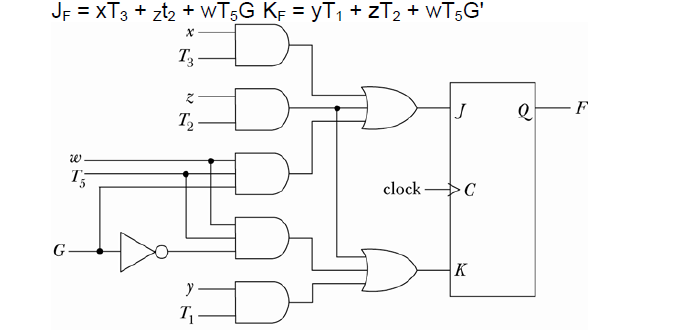
## سوال دوم(+)

عملیاتی که باید با فلیپ فلاپ F انجام شود (در کامپیوتر پایه استفاده نشده) با دستورات register transfer زیر مشخص می شود:

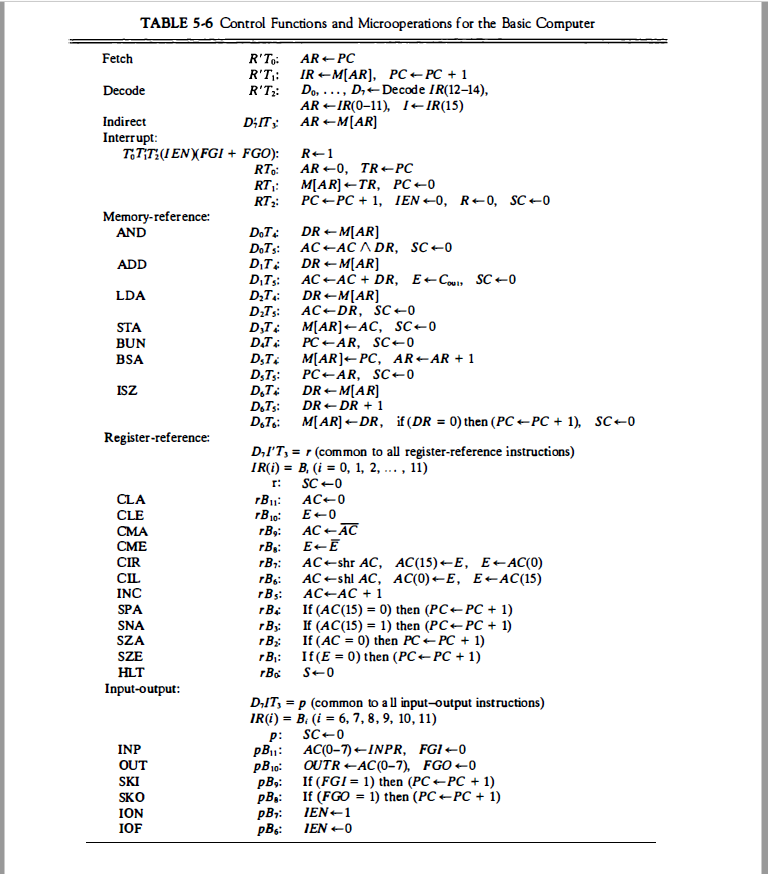


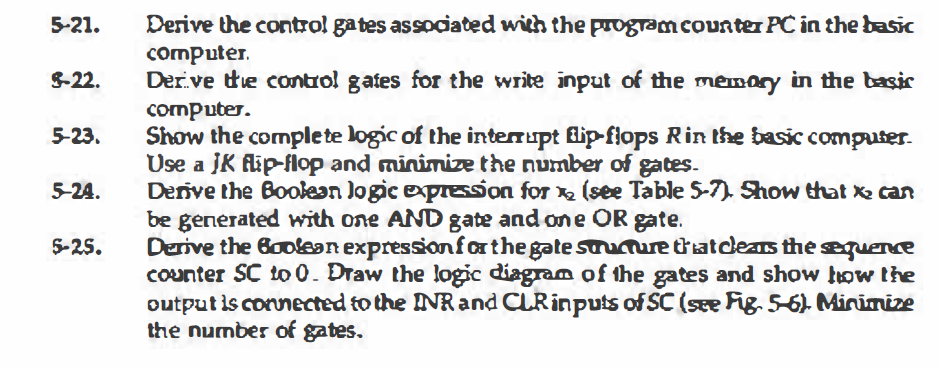
در غیر اینصورت محتوای F نباید تغییر کند. logic diagram را برای نشان دادن ارتباط میان گیت هایی که توابع کنترلی را میسازند و ورودی های فلیپ فلاپ رسم کنید. از JK فلیپ فلاپ استفاده کنید و تعداد گیت ها را کمینه کنید.

پاسخ:



## سوال سوم(نمونه سوال)





# Shayan

1- یک کامپیوتر پایه با ابعاد d \* 8 داریم. با توجه به ابعاد حافظه مشخص کنید هر یک از ثبات های AR، IR، PC، DR و AC باید چند بیتی باشند؟(+)

PC. AR → contain address = log2(d) =size

AC,DR → contain data = 8bit

IR = contain data = data size \* number of word = 8 \* w

~~2- انواع نحوه پیاده­سازی Control Units را نام برده و توضیح دهید~~

~~3- برای محاسبه دنباله کد A + B –C =Dدر هر یک از Stack Machine ,Accumulator Machine, Register-Memory Machine, Load-Store Machine دنباله عملیات های لازم را بنویسید~~

4- انواع مدل­های آدرس دهی را نام برده و توضیح دهید(امتحان)

<https://www.geeksforgeeks.org/addressing-modes/>

•Implied mode

Complement ACC

•Immediate Mode

Operand is inside the instruction

ADD R1, 105

•Register Mode

Operand is a register

Add R1, R2, R3

•Register Indirect Mode

Add R1, R2, [R3]

•Auto-increment/decrement Mode: Add R1, R2, [R3+4]

•Direct Mode

•Indirect Mode

•Relative Mode

Relative to PC

•Index Addressing Mode

•The content of an index reg. is added to address part of the instr: Add R1, R2, [R3+Const.]

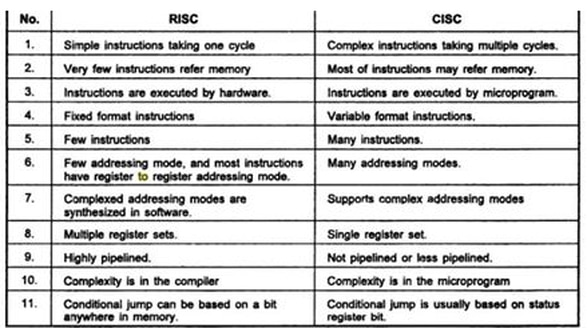
•

•Base Register Mode

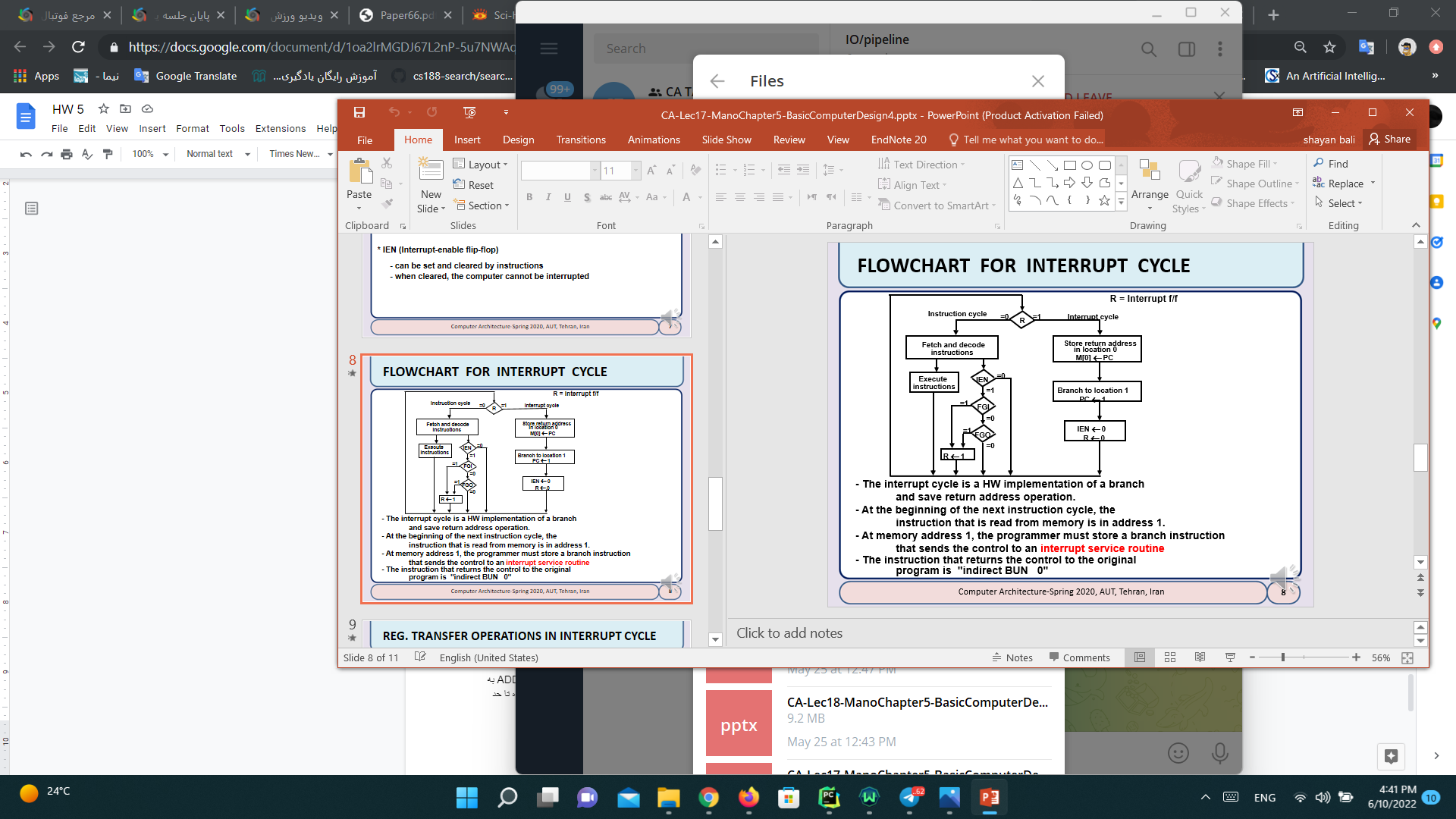
•The address part of the intr. Is added to the content of a base reg.

•LW R1, 100(R3)

~~5- تفاوت های کامپیوتر RISC, CISC را به طور کامل بنویسید(امتحان)~~

~~~~

6- بروز اینتراپت را در غالب یک فلوچارت مشخص کنید(امتحان)



# 

# 

# 

# Hedieh - Lecture 19

## سوال اول(+)

برای هر یک از حالت‌های زیر چه نوع معماری‌ای را برای طراحی پردازنده پیشنهاد می‌دهید. در هر مورد محدودیت‌ها مزایا و دلیل انتخاب آن را توضیح دهید.

الف) در یک کاربرد خاص، شرکتی نیاز به طراحی پردازنده‌ای دارد که دستورات پیچیده‌ای را انجام دهد اما در عین حال تا جد ممکن هزینه‌ی کمی را برای ساخت تجهیزات مورد نیاز خرج کند.

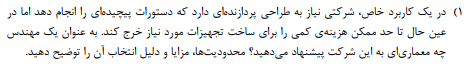
ب) در این مورد برای طراحی ریزپردازنده نیاز است بیشتر دستورات در یک پالس ساعت انجام پذیرند و فضای زیادی برای اختصاص به پشته وجود ندارد.

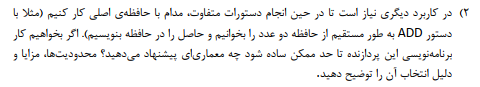
ج) در کاربرد دیگری نیاز است تا در حین انجام دستورات متفاوت مدام با حافظه‌ی اصلی کار کنیم. (مثلا با دستور ADD به طور مستقیم از حافظه دو عدد را بخوانیم و حاصل را در حافظه بنویسیم). اگر بخواهیم کار برنامه‌نویسی این پردازنده تا حد ممکن ساده شود.

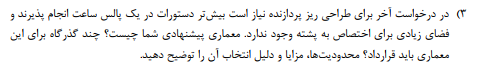
الف) معماری RISC پیشنهاد میشود. دستورات در معماری RISC نسبتا ساده اند و در این کاربرد هم به دستورات پیچیده نیاز نداریم پس میتوان از این معماری استفاده کرد. از طرفی چون تعداد ترانزیستورها در پردازنده ریسک کمتر است، هزینه ی کمتری برای ساخت تجهیزات مورد نیاز مصرف میشود.

ب) معماری RISC پیشنهاد میشود. مهمترین مشخصه پردازنده های (RISC) برخلاف CISC در این است که %۹5دستورات در یک سیکل ساعت اجرا میشوند. همه ریزپردازنده های RISC حداقل 3۲ثبات دارند که مزیت آن عدم احتیاج به یک پشته بزرگ برای دخیره پارامترهاست. پردازنده های RISC گذرگاه های جداگانه ای برای داده و کد دارند پس دو گذرگاه قرار میدهیم.

ج) معماری CISC پیشنهاد میشود. در RISC دسترسی به حافظه تنها از طریق دستورالعملهای خاصی قابل انجام است و به عنوان مثال نمیتوان از بخشی از دستور add به حافظه دسترسی داشت. پردازنده های RISC مجموعه کوچکی از دستورات شامل دستورات پایه ای را در خود دارند که کار برنامه نویسان اسمبلی را در مقایسه با اسمبلی نویسی برای پردازنده های CISC سخت میکند.







## سوال دوم

~~دستورات(code sequence) مربوط به تکه کد زیر را در هر یک از حالت های stack machine, accumulator machine و load-store machine بنویسید.~~

~~D = C \* ( A + B )~~