نمونه سوالات معماری کامپیوتر دکتر زینالی

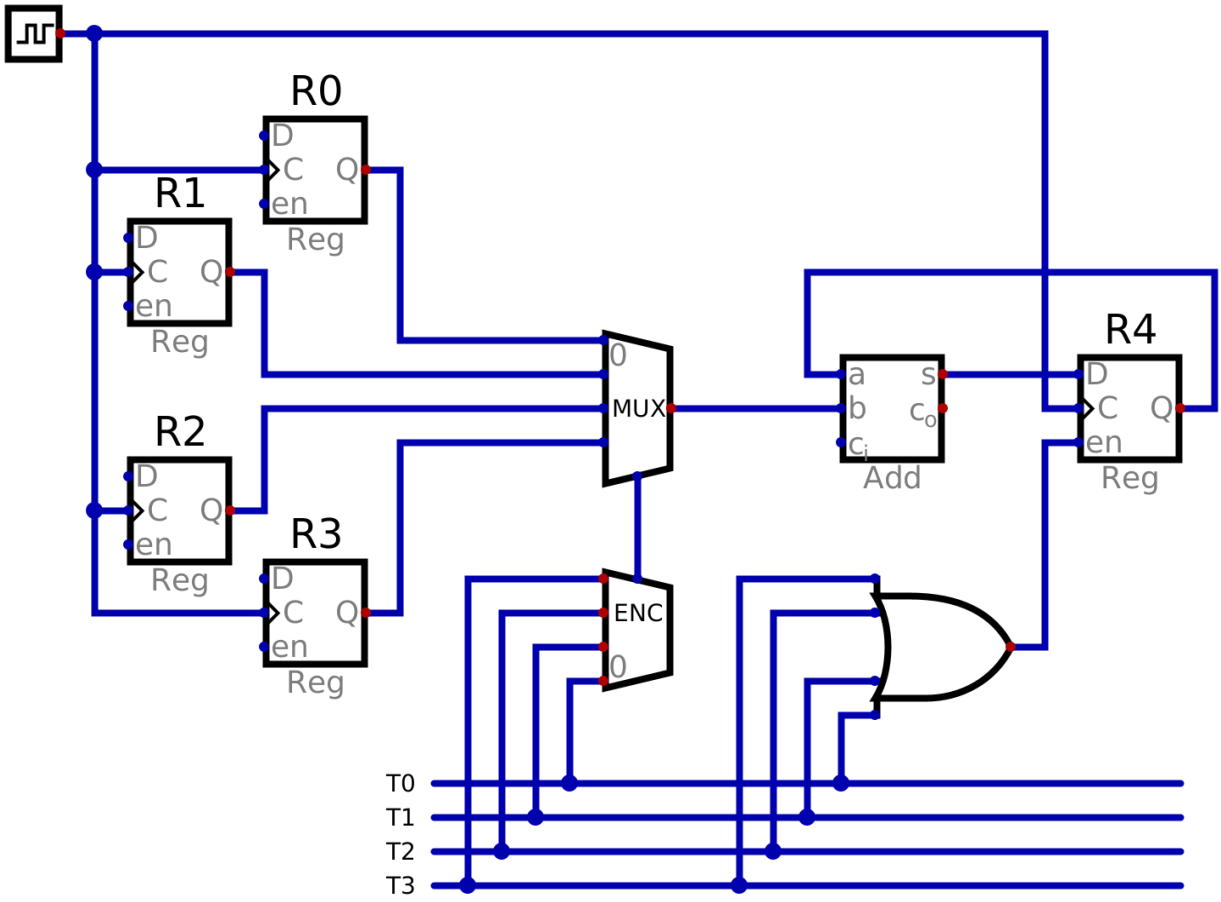
جمع آوری شده کلاس حل تمرین رامتین کوثری

RTL and Block Diagram - بلوک دیاگرام ها و آر تی ال

Question 0 - *Class Example*

$$T_0 : \quad R_4 \leftarrow R_4 + R_0$$
$$T_1 : \quad R_4 \leftarrow R_4 + R_1$$
$$T_2 : \quad R_4 \leftarrow R_4 + R_2$$
$$T_3 : \quad R_4 \leftarrow R_4 + R_3$$

Solution :



Question 1 - *Mordad 1400 Mid-Term*

zT_1 : $if(R_1 = 0)\ then$

$R_1 \leftarrow 0$

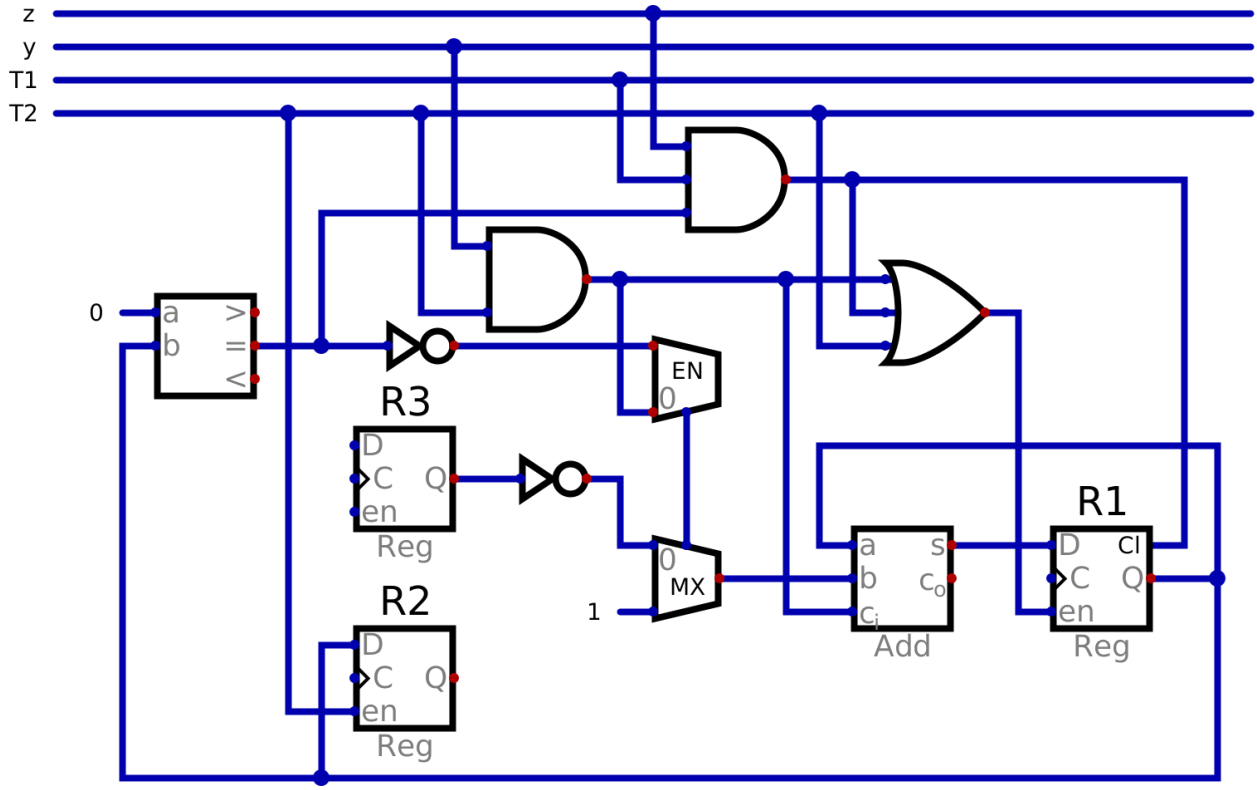
$else$

$R_1 \leftarrow R_1 + 1$

T_2 : $R_2 \leftarrow R_1$

yT_2 : $R_1 \leftarrow R_1 - R_3$

Solution :



Question 2 - *Dey 1400 Final Exam*

PQ' : $M[R_2] \leftarrow R_1$

PQ : $if(R_1 = R_2)\ then$

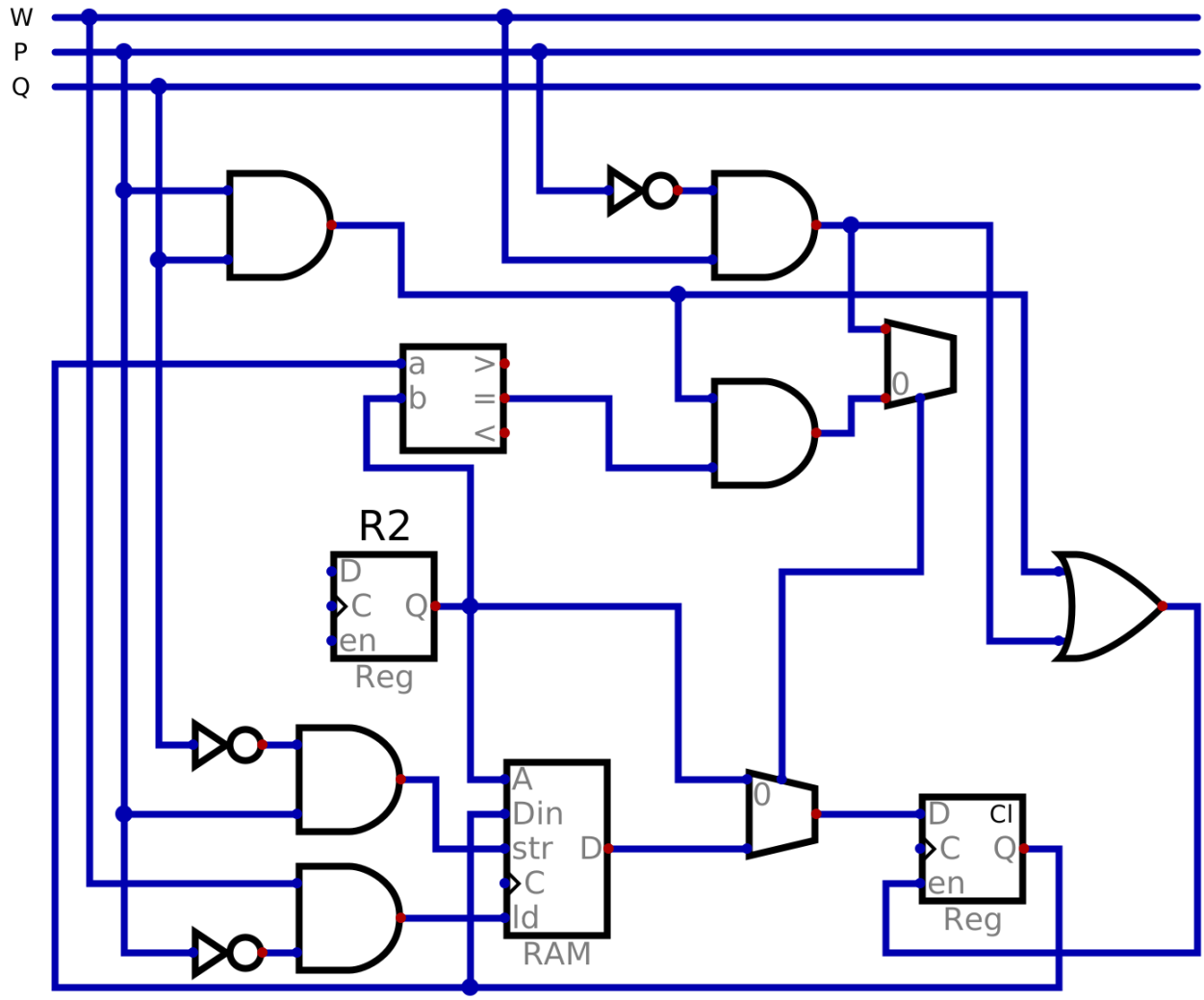
$R_1 \leftarrow R_2$

$else$

$R_1 \leftarrow 0$

$P'W$: $R_1 \leftarrow M[R_2]$

Solution :



Question 3 *- Dey 1401, Tir 1402, Tir 1403 Final Exams, Mordad 1401 Mid-term*

xT_0 : $if(R_1 = R_2)\ then$

$R_1 \leftarrow R_1 + 1$

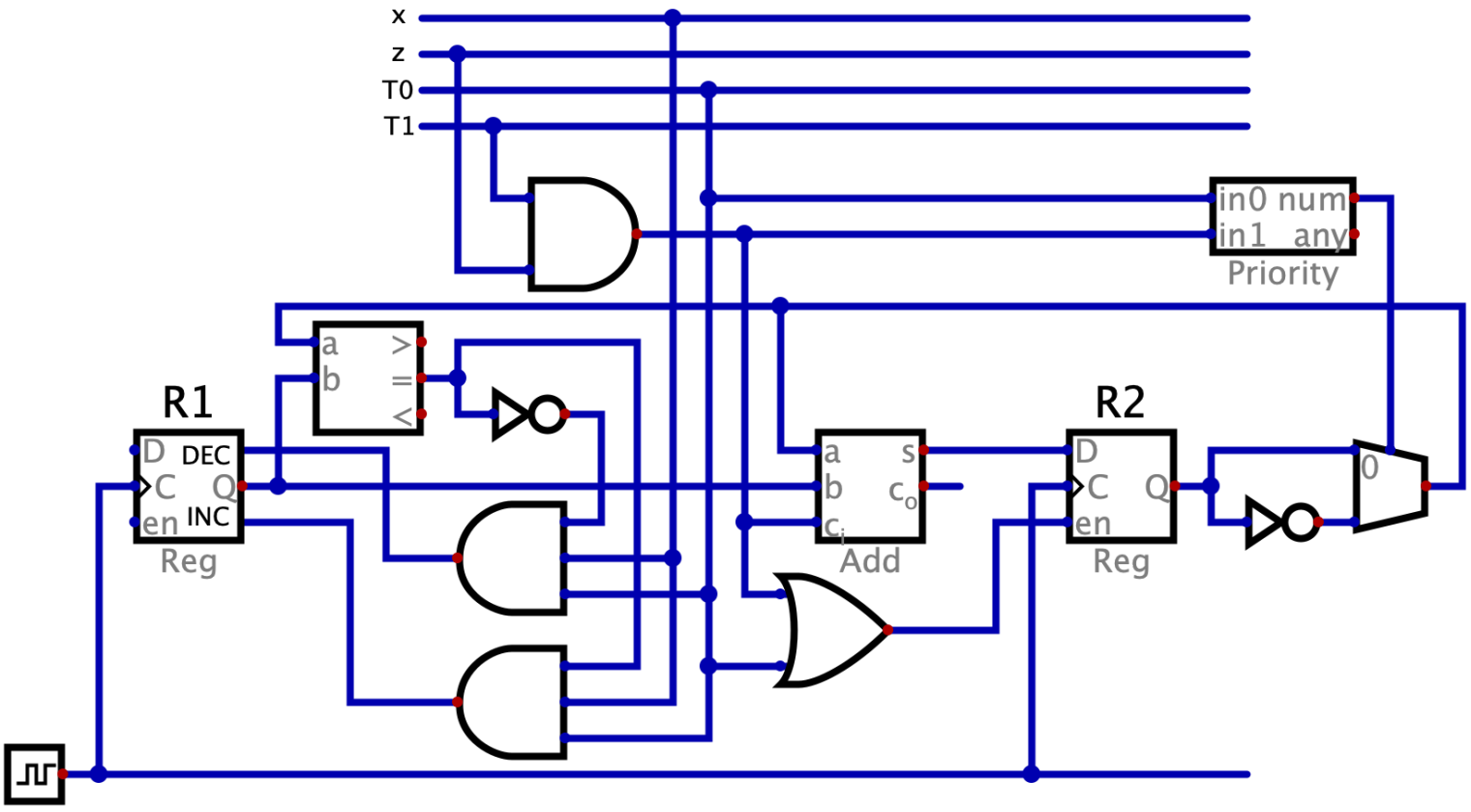
$else$

$R_1 \leftarrow R_1 - 1$

T_0 : $R_2 \leftarrow R_1 + R_2$

zT_1 : $R_2 \leftarrow R_1 - R_2 = R_1 + R_2' + 1$

Solution :



Question 4 *- Azar 1400 Mid-term*

xT_0 : $if(R_1 = R_2)\ then$

$R_1 \leftarrow R_1 + 1$

$else$

$R_1 \leftarrow R_1 - 1$

T_2 : $R_1 \leftarrow R_1 + R_2$

zT_1 : $R_1 \leftarrow R_1 - R_2$

Solution :

Question 5 *- Mordad 1403 Mid-term*

$P'W$: $R_1 \leftarrow M[R_2]$

PQ' : $M[R_2] \leftarrow R_1$

PQ : $if(R_1 = M[R_2])\ then$

$R_1 \leftarrow 0$

$else$

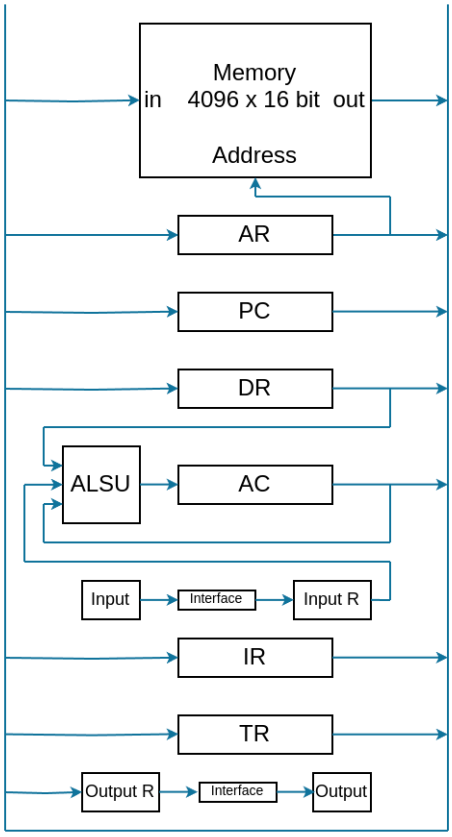
$R_1 \leftarrow R_1 + 1$

Solution :

DPU Instructions RTL and Hardwire - و نوشتن آر تی ال دستورات
طراحی واحد دیتا به روش هارد وایر

Question 6 - *Class Example*

I	Opcode	Address
1	3	12



D_0	AND	000	$AC \leftarrow AC \wedge M[AR]$
D_1	ADD	001	$AC \leftarrow AC + M[AR]$
D_2	LDAC	010	$AC \leftarrow M[AR]$
D_3	STAC	011	$M[AR] \leftarrow AC$
D_4	BUN	100	$PC \leftarrow AR$
D_5	BSA	101	$M[AR] \leftarrow PC\,,\ PC \leftarrow PC + 1$
D_6	ISZ	110	$M[AR] \leftarrow M[AR] + 1\,,\ if\ (M[AR+1] = 0)\ then\ Skip$

نکته ۱ : Skip یعنی افزایش دادن پروگرم کانتر : $PC \leftarrow PC + 1$

نکته ۲ : برای نوشتن RTL شرط مساوی بودن از زیرو فلگ **(z)** استفاده میکنیم :

$$for\ Register: \quad if\ Reg = 0 \Rightarrow z = 1 \quad and \quad if\ Reg \neq 0 \Rightarrow z = 0$$

نکته ۳ : در $AC \leftarrow M[AR]$ عمل Transfer اتفاق می افتد که توسط ALSU انجام میشود
ولی چون ALSU در خودش Transfer ندارد، میتوانیم این عمل را به ۲ شکل انجام دهیم :

$$AC \leftarrow M[AR] \equiv AC \leftarrow M[AR] \vee M[AR] \equiv AC \leftarrow M[AR] + 0$$

نکته ۴ : با توجه فرمت دستور، هر دستور ۱۶ بیت را اشغال کرده پس در هر خانه مموری فقط ۱ دستور وجود دارد.

Solution :

ابتدا مراحل گفته شده را انجام میدهیم :

1. **عددگذاری مالتی پلکسر ها :** از آنجایی که ما یک باس داریم که ۷ رجیستر و مموری به آن وصل شده اند پس مالتی پلکسر را از صفر الی ۷ مقدار دهی می کنیم که حال صفر را چون استفاده نمیکنیم حالت Idle می نامیم.
   سپس بازه بیت ها را در فرمت دستور شخص میکنیم، برای مثال بیت ۱۵ ام باید در I ریخته شود.
2. **رجیستر IR :** خوشبختان رجیستر IR در اینجا وجود دارد پس فقط از همان استفاده میکنیم.
3. **عددگذاری پیشوند های Opcode ها :** همانطور که در جدول میبینید با توجه به ۳ بیت داده شده دستورات را از D_0 الی D_7 عدد گذاری کرده ایم.
4. **نوشتن RTL دستورات :** برای نوشتن RTL های دستورات باید تمامی مراحل چرخه دستورات را بنویسیم :

Fetch : $IR \leftarrow M[PC]\ ,\ PC \leftarrow PC + 1$

T_0 : $AR \leftarrow PC$ — چون می خواهیم از پروگرم کانتر استفاده کنیم و مستقیم به

T_1 : $IR \leftarrow M[AR]\ ,\ PC \leftarrow PC + 1$ — مموری وصل نیست از آدرس رجیستر به عنوان واسطه استفاده میکنیم

همانطور که می بینیم این مرحله ۲کلاک طول کشیده که انجام شود. همچنین توجه داشته باشید که از پایه INC برای اضافه کردن PC استفاده کرده ایم.

Decode : $I \leftarrow IR_{(15)}\ ,\ D_0D_1D_2 \dots D_7 \leftarrow IR_{(14-12)}\ ,\ AR \leftarrow IR_{(11-0)}$

$T_2:\quad I \leftarrow IR_{(15)}\ ,\ D_0D_1D_2 \dots D_7 \leftarrow IR_{(14-12)}\ ,\ AR \leftarrow IR_{(11-0)}$ این مرحله خودش در ۱ کلاک انجام می شود

Effective Address : $if\ (I = 1)\ then\ AR \leftarrow M[AR]$

$I'T_3:\quad NOP$ **نکته ۵ :** دستورات شرطی را با فلگ ها نمایش می دهیم

$IT_3:\quad AR \leftarrow M[AR]$ **نکته ۶ :** چون خط اول کاری انجام نمیدهد میتوانیم آن را اصلا ننویسیم

Execute : $D_0D_1D_2 \dots D_7$ **نکته ۷ :** این مرحله باید برای تمامی دستورات نوشته شود ولی بدنه مشترک خیر

$D_0(AND):\quad AC \leftarrow AC \wedge M[AR]$

$D_0T_4:\quad DR \leftarrow M[AR]$ **نکته ۸ :** در مرحله اجرا زمان ها باید با دستورات مربوطه 'اند' شوند زیرا در یک زمان فقط یکی از دستورات باید باشد

$D_0T_5:\quad AC \leftarrow AC \wedge DR\ ,\ SC \leftarrow 0$ **نکته ۹ :** برای جلوگیری از هدر رفت کلاک سیکونسر را صفر میکنیم تا از واکشی شروع کند

$D_1(ADD):\quad AC \leftarrow AC + M[AR]$

$D_1T_4:\quad DR \leftarrow M[AR]$

$D_1T_5:\quad AC \leftarrow AC + DR\ ,\ SC \leftarrow 0$

$D_2(LDAC):\quad AC \leftarrow M[AR]$

$D_2T_4:\quad DR \leftarrow M[AR]$

$D_2T_5:\quad AC \leftarrow DR\ ,\ SC \leftarrow 0$

$D_3(STAC):\quad M[AR] \leftarrow AC$

$D_3T_4:\quad M[AR] \leftarrow AC\ ,\ SC \leftarrow 0$

$D_4(BUN):\quad PC \leftarrow AR$

$D_4T_4:\quad PC \leftarrow AR\ ,\ SC \leftarrow 0$

$D_5(BSA):\quad M[AR] \leftarrow PC\ ,\ PC \leftarrow AR + 1$

$D_5T_4:\quad M[AR]_{(11-0)} \leftarrow PC\ ,\ AR \leftarrow AR + 1$ **نکته ۱۰ :** چون پروگرم کانتر ۱۲ بیت است آن را فقط در ۱۲ بیت آدرس بلاک حافظه میریزیم

$D_5T_5:\quad AR \leftarrow PC\ ,\ SC \leftarrow 0$ **نکته ۱۱ :** برای اضافه کردن آدرس رجیستر از پایه اینکریمنت استفاده کرده ایم نه ALSU

$D_6(ISZ):\quad M[AR] \leftarrow M[AR] + 1\ ,\ if\ (M[AR + 1] = 0)\ then\ Skip$

$D_6T_4:\quad DR \leftarrow M[AR]$ **نکته ۱۲ :** ممووری پایه اینکریمنت ندارد و باید یا در یک رجیستر بریزیم اینکریمنت کنیم یا با ALSU

$D_6T_5:\quad AC \leftarrow DR + 1$

$z\,D_6T_6:\quad M[AR] \leftarrow AC\ ,\ PC \leftarrow PC + 1\ ,\ SC \leftarrow 0$

$z'D_6T_6:\quad M[AR] \leftarrow AC\ ,\ SC \leftarrow 0$

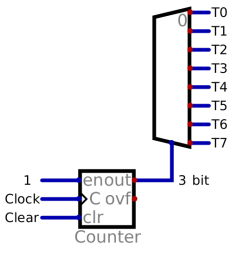
نکته ۱۳ : از آنجا که ۲ عبارت $M[AR] \leftarrow AC$ و $SC \leftarrow 0$ تکرار شده اند می توانیم آنها را فاکتور بگیریم و به صورت کلی در D_6T_5 انجام دهیم :

$z\,D_6T_6:\quad PC \leftarrow PC + 1$

$D_6T_6\ :\quad M[AR] \leftarrow AC\ ,\ SC \leftarrow 0$

5. رسم Sequencer : پس از رسم RTL ها درمیابیم که بیشترین زمانی که طی شده تا دستور انجام شود T_6 است.

حال این ۷ حالت زمانی را با چند عدد میتوانیم کد کنیم ؟ با ۳ بیت زیرا $۲^۳ = ۸$ پس داریم :



6. **پایه ها :** حال باید بررسی کنیم که پایه های مموری و رجیستر ها در چه زمان هایی فعال می شوند، به صورت کلی با بررسی DPU به پایه های زیر میرسیم :

$Read : T_1 + IT_3 + D_0T_4 + D_1T_4 + D_2T_4 + D_6T_4$

$Write : D_3T_4 + D_5T_4 + D_6T_6$

$Clear(SC) : D_0T_5 + D_1T_5 + D_2T_5 + D_3T_4 + D_4T_4 + D_5T_5 + D_6T_6$

$LD(AR) : T_0 + T_2 + IT_3$

$LD(IR) : T_1$ توجه داشته باشید که اگر المان ما سمت چپ باشد در آن چیزی ریخته میشود ولی اگر راست باشد یعنی از آن خوانده میشود

$LD(DR) : D_0T_4 + D_1T_4 + D_2T_4 + D_6T_4$

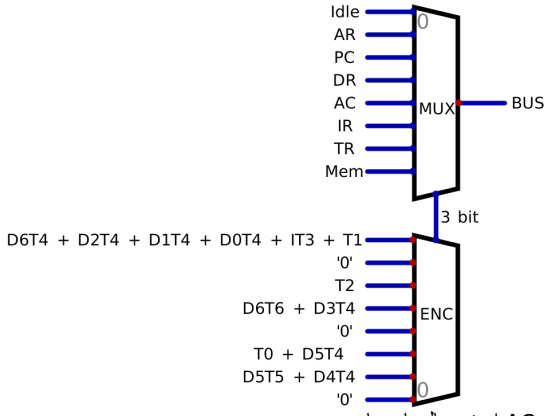
$LD(PC) : T_4 + D_5T_5$

$LD(AC) : D_0T_5 + D_1T_5 + D_2T_5 + D_6T_5$

$INC(AR) : D_5T_4$

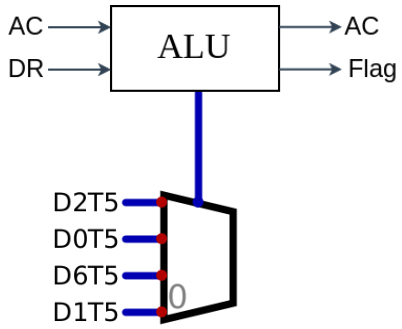
$INC(PC) : T_1 + zD_6T_6$

7. **طراحی Bus :** در این DPU ما تنها ۱ باس داریم که ۷ عدد رجیستر و مموری به آن متصل است که به یکدیگر راه دارند و از آنجایی که باس را با مالتی پلکسر طراحی میکنند، میتوانیم یک مالتی پلکسر با ۳ بیت سلکت و یک انکودر با ۸ ورودی داشته باشیم که طراحی آن به شکل زیر امکان پذیر است :



نکته ۱۴ : توجه داشته باشید که ورودی های مالتی پلکسر ماژول های ما میباشد مانند رجیستر ها و مموری، همچنین ورودی های انکودر ما زمانبندی هایی هستند که در آنها یک و فقط یک رجیستر بخصوص یا مموری فعال میشود. همچنین توجه کنید که ما در مرحله ۶ پایه LD برای AC را نوشتیم ولی در باس به آن مقدار صفر را دادیم زیرا AC به باس مستقیم وصل نیست.

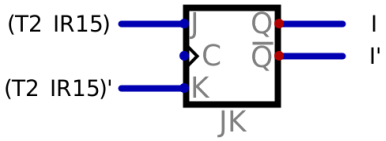
8. **دستورات ALSU :** باید توجه داشته باشیم که مقصد همه دستورات ALSU رجیستر AC است. حال باید بیاییم و در RTL های نوشته شده ببینیم که کجا از ALSU استفاده کرده ایم و سپس آن را طراحی کنیم :



C_1	C_0	AC
0	0	$AC + DR$
0	1	$AC + 1$
1	0	$AC \wedge DR$
1	1	$DR \vee$

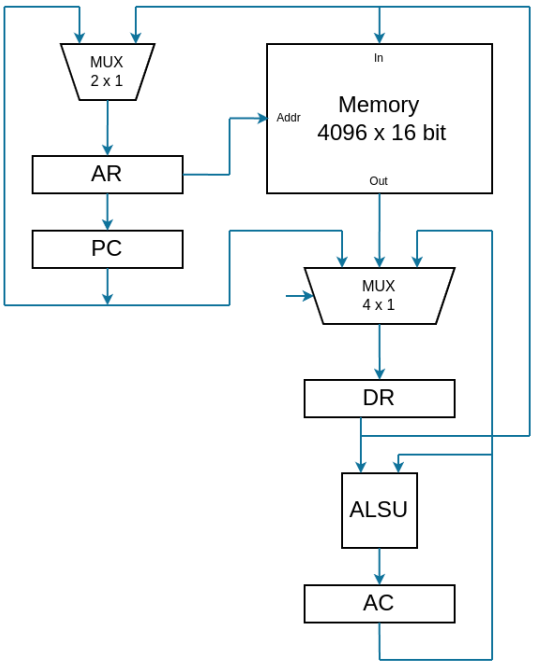
نکته ۱۵ : توجه داشته باشید که چون فقط ۲ نوع کاربرد از ALSU داریم پس آن را ALU نوشتیم. همچنین باید زمانبندی های مربوطه را به ALU به وسیله انکودر متصل کنیم.

9. **طراحی فلگ ها :** فلگ ها را با $JK\ f.f$ (جی کا فلیپ فلاپ) طراحی میکنیم. در DPU داده شده یک فلگ I داریم و فلگ z که فلگ z یک فلگ خروجی است که از رجیستر بیرون می آید و ما لازم نیست آن را مقداردهی کنیم خود رجیستر مقداردهی می کند پس فقط فلگ I را طراحی میکنیم :

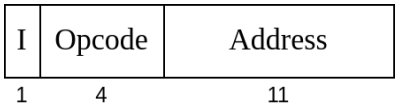


نکته ۱۶ : توجه داشته باشید که فقط در زمان T_2 هنگامی که داشتیم مرحله دیکد را انجام میدادیم فقط بیت ۱۵ ام رجیستر IR را در فلگ I ریختیم پس کافیست زمان را با IR_{15} اند کرده و به ورودی J بدهیم و ناتش را به ورودی K.

Question 7 - *Class Example*



D_0	ADD	0000	$AC \leftarrow AC + M[AR]$
D_1	Branch	0001	$if\ (AC < 0)\ then\ PC \leftarrow AR$
D_2	Store	0010	$M[AR] \leftarrow AC$
D_3	Exchange	0011	$AC \leftarrow M[AR]\ ,\ M[AR] \leftarrow AC$



Solution :

ابتدا مراحل گفته شده را انجام میدهیم :

1. **عددگذاری مالتی پلکسر ها :** ۲ باس داریم، مانند سوال قبل عدد گذاری میکنیم. فرمت دستور ما نیز مانند سوال قبل است تنها تفاوت این است که Opcode ۴ بیتی است یعنی ۱۶ دستور را میتوانیم در این ماشین داشته باشیم.
2. **رجیستر IR :** رجیستر IR در اینجا وجود ندارد پس باید ببینیم از کدام رجیستر میتوانیم به عنوان رجیستر دستور استفاده کنیم. با کمی دقت در میابیم که داده وقتی از مموری خارج می شود مستقیم به رجیستر DR میرود پس میتوانیم از همان به عنوان رجیستر دستور استفاده کنیم.
3. **عددگذاری پیشوند های Opcode ها :** همانطور که در جدول میبینید با توجه به ۳ بیت داده شده دستورات را از D_0 الی D_{15} عدد گذاری کرده ایم.
4. **نوشتن RTL دستورات :** برای نوشتن RTL های دستورات باید تمامی مراحل چرخه دستورات را بنویسیم :

Fetch : $IR \leftarrow M[PC]\ ,\ PC \leftarrow PC + 1$

$T_0 :\quad AR \leftarrow PC$ چون می خواهیم از پروگرم کانتر استفاده کنیم و مستقیم به مموری وصل نیست

$T_1 :\quad DR \leftarrow M[AR]\ ,\ PC \leftarrow PC + 1$ از آدرس رجیستر به عنوان واسطه استفاده میکنیم. همچنین از دیتا رجیستر باید استفاده کرد

همانطور که می بینیم این مرحله ۲کلاک طول کشیده که انجام شود. همچنین توجه داشته باشید که از پایه INC برای اضافه کردن PC استفاده کرده ایم.

Decode : $I \leftarrow IR_{(15)}\ ,\ D_0D_1D_2 \dots D_7 \leftarrow IR_{(14-12)}\ ,\ AR \leftarrow IR_{(11-0)}$ از دیتا رجیستر باید استفاده کنیم

$T_2 :\quad I \leftarrow DR_{(15)}\ ,\ D_0D_1D_2 \dots D_{15} \leftarrow DR_{(14-11)}\ ,\ AR \leftarrow DR_{(10-0)}$ این مرحله خودش در ۱ کلاک انجام می شود

Effective Address : $if\ (I = 1)\ then\ AR \leftarrow M[AR]$

$I'T_3 :\quad NOP$

$IT_3 :\quad DR \leftarrow M[AR]$ چون مموری به رجیستر آدرس مستقیم متصل نیست باید از رجیستر دیتا به عنوان واسط استفاده کنیم

$IT_4 :\quad AR \leftarrow DR$

Execute : $D_0D_1D_2 \dots D_7$

$D_0(ADD) :\quad AC \leftarrow AC + M[AR]$

$D_0T_5: \quad DR \leftarrow M[AR]$

$D_0T_6: \quad AC \leftarrow AC + DR \ , \ SC \leftarrow 0$

$D_1(Branch): \quad if\ (AC < 0)\ then\ PC \leftarrow AR$

نکته ۱۷ : ما برای پیاده سازی $AC < 0$ میتوانیم از فلگ های s یا sign و z یا zero استفاده کنیم، فلگ s زمانی فعال است که عدد رجیستر ما علامت دار، یعنی منفی باشد و فلگ z زمانی فعال است که عدد رجیستر ما صفر باشد. حال ما میتوانیم حالت های مقایسه عدد رجیستر را با صفر به شکل های زیر بنویسیم :

$AC < 0$	منفی باشد	s
$AC \leq 0$	منفی باشد یا صفر باشد	$s + z$
$AC > 0$	مثبت باشد و صفر نباشد	sz'
$AC \geq 0$	مثبت باشد (نکته زیر جدول)	s'

توجه داشته باشید که عدد صفر شامل دسته اعداد مثبت میشود به همین دلیل تاثیر این جمله را در جدول میبینید. همچنین با توجه به جدول و RTL دستور Branch فقط به $AC < 0$ نیاز داریم که کافیست تنها از فلگ s استفاده کنیم تا معادل آن پیاده سازی شود.

نکته ۱۸ : با توجه به اینکه فلگ های z و s فلگ های خروجی از رجیستر AC هستند پس نیازی نیست آنها را در مرحله ۹ طراحی کنیم.

حال که مشخص شد چگونه باید از این فلگ ها استفاده کنیم میتوانیم RTL مربوطه مرحله اجرای این دستور را بنویسیم :

نکته ۱۹ : ما می توانیم RTL این مرحله را به شکل زیر بنویسیم :

$sD_1T_5: \quad PC \leftarrow AR \ , \ SC \leftarrow 0$

ولی حواستان باشد که اگر به این شکل بنویسید، SC فقط هنگامی که فلگ s فعال باشد صفر میشود و اینطوری باعث میشود که اگر فلگ s غیرفعال بود SC صفر نشده و Sequencer همینطور زمانبندی های بعدی را انجام دهد، بنابراین ما باید مرحله صفر کردن Sequencer را خارج از شرایط فلگ ولی در **همان کلاک** انجام دهیم (مانند فاکتور گرفتن در سوال قبل) :

$sD_1T_5: \quad PC \leftarrow AR$

$D_1T_5: \quad SC \leftarrow 0$

$D_2(Store): \quad M[AR] \leftarrow AC$

$D_2T_5: \quad DR \leftarrow AC$

$D_2T_6: \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 0$

$D_3(Exchange): \quad AC \leftarrow M[AR] \ , \ M[AR] \leftarrow AC$

نکته ۲۰ : ما میتوانیم ابتدا دستورات را به شکل زیر در ۴ کلاک بنویسیم :

$D_3T_5: \quad DR \leftarrow M[AR]$

$D_3T_6: \quad AC \leftarrow DR$

$D_3T_7: \quad DR \leftarrow AC$

$D_3T_8: \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 0$

ولی این امر امکان پذیر نیست زیرا در زمان های T_6 و T_7 اگر در ۲ کلاس جدا آنها را انجام دهیم انگار اتفاقی نیفتاده است زیرا DR در آخر همان مقدار اولیه خودش را میگیرد. پس باید این ۲ دستور را موازی و همزمان انجام دهیم (اگر شرایطش بود و از لحاظ سخت افزاری قابلیت موازی اجرا شدن طراحی شده بود که در این مسئله شده است) به این صورت :

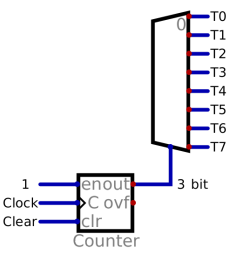
$D_3T_5: \quad DR \leftarrow M[AR]$

$D_3T_6: \quad AC \leftarrow DR \ , \ DR \leftarrow AC$

$D_3T_7: \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 0$

5. رسم **Sequencer :** پس از رسم RTL ها درمیابیم که بیشترین زمانی که طی شده تا دستور انجام شود T_7 است

حال این ۸ حالت زمانی را با چند عدد میتوانیم کد کنیم ؟ با ۳ بیت زیرا $۸ = ۲^۳$ پس داریم :



6. **پایه ها :** حال باید بررسی کنیم که پایه های ممووری و رجیستر ها در چه زمان هایی فعال می شوند، به صورت کلی با بررسی DPU به پایه های زیر میرسیم :

$Read : T_1 + IT_3 + D_0T_5 + D_3T_5$

$Write : D_2T_6 + D_3T_7$

$Clear(SC) : D_0T_6 + D_1T_5 + D_2T_6 + D_3T_7$

$LD(AR) : T_0 + T_2 + IT_4$

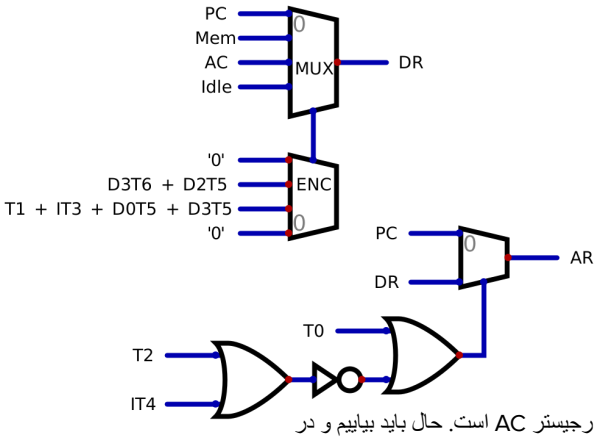
$LD(DR) : T_1 + IT_3 + D_0T_5 + D_2T_5 + D_3T_6$

$LD(PC) : sD_1T_5$

$LD(AC) : D_0T_6 + D_3T_6$

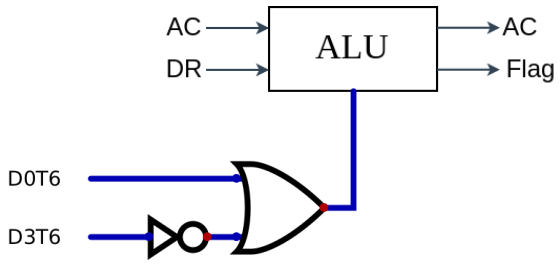
$INC(PC) : T_1$

7. **طراحی Bus :** در این DPU ما ۲ باس داریم که به یکی ۲ المان و به دیگری ۳ المان متصل است که به یکدیگر راه دارند و از آنجایی که باس را با مالتی پلکسر طراحی میکنند، میتوانیم یک مالتی پلکسر با ۲ بیت سلکت و یک انکودر با ۴ ورودی داشته باشیم و همچنین یک مالتی پلکسر با ۱ بیت سلکت بدون هیچ انکودری برای آن زیرا می توانیم تنها با ۱ بیت سلکت کنیم که طراحی آن به شکل زیر امکان پذیر است :

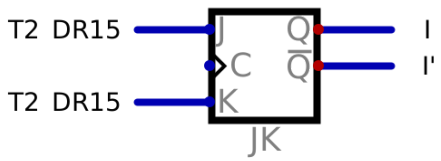


نکته ۲۱ : توجه داشته باشید که به جای پیاده سازی انکودر در مالتی پلکسر دوم توانستیم آن را با گیت های منطقی به شکل ساده تر پیاده سازی کنیم.

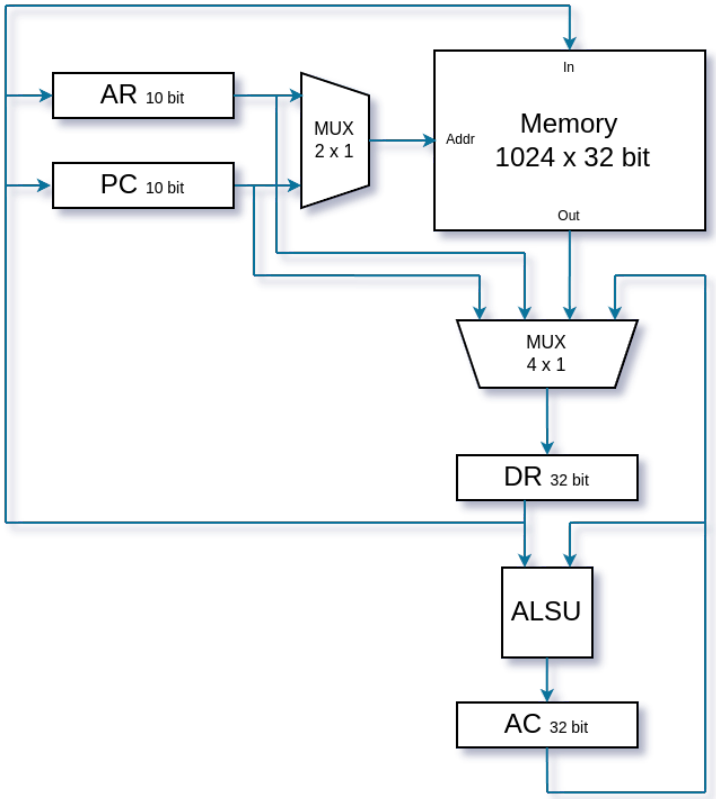
8. **دستورات ALSU :** باید توجه داشته باشیم که مقصد همه دستورات ALSU رجیستر AC است. حال باید بیاییم و در RTL های نوشته شده ببینیم که کجا از ALSU استفاده کرده ایم و سپس آن را طراحی کنیم :



9. **طراحی فلگ ها :** فلگ ها را با $JK\ f.f$ (جی کا فلیپ فلاپ) طراحی میکنیم. در DPU داده شده یک فلگ I داریم و فلگ s که فلگ s یک فلگ خروجی است که از رجیستر بیرون می آید و ما لازم نیست آن را مقداردهی کنیم خود رجیستر مقداردهی می کند پس فقط فلگ I را طراحی میکنیم.



Question 8 - *Dey 1400, Tir 1403 Final Exam, Dey 1402 Mid-term*



D_3	XNOR	00011	$AC \leftarrow (AC \oplus M[AR])'$
D_{12}	SUBM	01100	$M[AR] \leftarrow AC - M[AR]$
D_{15}	ADDM	01111	$M[AR] \leftarrow AC + M[AR]$

I_1	$Opcode_1$	$Address_1$	I_2	$Opcode_2$	$Address_2$
1	5	10	1	5	10

Direct : I = 1
Indirect : I = 0

نکته ۲۱ : همانطور که در شکل میبینیم PC به حافظه به صورت مستقیم متصل است پس می توانیم مرحله Fetch را با یک کلاک انجام دهیم.

نکته ۲۲ : داده وقتی از حافظه بیرون می آید مستقیم به DR میرود و با توجه به شکل، DR ظرفیت ۳۲ بیتی دارد پس کل یک خانه حافظه (که در اینجا شامل ۲ دستور میباشد) درون این رجیستر قرار میگیرد. ولی اگر DR ما مثلا ۱۶ بیت بود، در مرحله Fetch یکبار باید دستور اول و بار دیگر باید دستور دوم را در DR ذخیره میکردیم، یعنی در ۱ کلاک ولی در ۲ حالت (شرط) متفاوت این کار انجام می شد.

Solution :

ابتدا مراحل گفته شده را انجام میدهیم :

1. **عددگذاری مالتی پلکسر ها :** ۲ باس داریم، مانند سوال قبل عدد گذاری میکنیم. فرمت دستور نیز در مجموع ۳۲ بیت است که همان ۳۲ بیت از داخل حافظه بیرون می آید.
2. **رجیستر IR :** رجیستر IR در اینجا وجود ندارد پس باید ببینیم از کدام رجیستر میتوانیم به عنوان رجیستر دستور استفاده کنیم. با کمی دقت در میابیم که داده وقتی از مموری خارج می شود مستقیم به رجیستر DR میرود پس میتوانیم از همان به عنوان رجیستر دستور استفاده کنیم. توجه داشته باشید که این رجیستر ۳۲ بیت است. با توجه به نکته ۲۲ اگر این رجیستر ۱۶ بیت بود باید یکبار بخش اول فرمت دستور و بار دیگر بخش دوم را Fetch میکردیم.
3. **عددگذاری پیشوندهای Opcode ها :** همانطور که در جدول میبینید با توجه به ۵ بیت داده شده دستورات را از D_0 الی D_{31} عدد گذاری کرده ایم که در اینجا فقط دستورات D_3، D_{12} و D_{15} استفاده شده اند.
4. **نوشتن RTL دستورات :** برای نوشتن RTL های دستورات باید تمامی مراحل چرخه دستورات را بنویسیم :

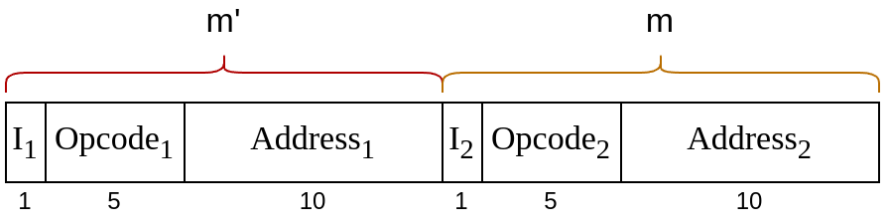
Fetch : $IR \leftarrow M[PC]$, $PC \leftarrow PC + 1$

T_0 : $DR \leftarrow M[PC]$, $PC \leftarrow PC + 1$ زیرا پروگرم کانتر مستقیم به مموری وصل است

همانطور که می بینیم این مرحله فقط ۱ کلاک طول کشیده که انجام شود. همچنین توجه داشته باشید که از پایه INC برای اضافه کردن PC استفاده کرده ایم.

Decode : در این مرحله خانه حافظه ما ۲ دستور دارد پس در ۲ حالت باید دستور را دیکد کنیم

برای حل این بخش ما نیاز داریم که فرمت دستور را ۲ بخش کنیم که یکبار بخش اول و بار دیگر بخش دوم را پردازش کنیم، در نتیجه ۲ حالت داریم پس این ۲ حالت را با ۱ بیت میتوانیم کد کنیم. نام این بیت را m میذاریم که اگر صفر بود بخش اول و اگر ۱ بود بخش دوم را پردازش کند :



حال کافی است با توجه به m' و m، مراحلی که نیاز است این فلگ کمکی را تاثیر دهیم. وقتی میخواهیم دستور را دیکد کنیم، یک بار باید Opcode 1 و بار دیگر باید Opcode 2 را دیکد کنیم پس باید برای اینکه مشخص کنیم آپکد کدام بخش را دیکد کنیم، از فلگ کمکی استفاده میکنیم :

$$m'T_1: \quad I \leftarrow DR_{(31)} \ , \ D_0D_1D_2 \dots D_{31} \leftarrow DR_{(30-26)} \ , \ AR \leftarrow DR_{(25-16)}$$
$$m\,T_1: \quad I \leftarrow DR_{(15)} \ , \ D_0D_1D_2 \dots D_{31} \leftarrow DR_{(14-10)} \ , \ AR \leftarrow DR_{(9-0)}$$

این مرحله همانطور که میبینید با ۱ کلاک انجام شده است ولی به ۲ حالت متفاوت که 0 یا 1 بودن فلگ کمکی مشخص میکند کدام خط اجرا شود.

Effective Address : $if\ (I = 0)\ then\ AR \leftarrow M[AR]$

در این مرحله اگر I برابر بود با ۱ یعنی **آدرس مستقیم** است پس **نباید کاری انجام دهیم** چه در بخش اول باشیم چه در بخش دوم یعنی داریم :

$$m'I\,T_2: \quad NOP$$
$$m\,I\,T_2: \quad NOP$$

نکته ۲۳ : همانطور که میبینید ۲ خط بالا در یک زمان اند و خروجی هر ۲ یکسان ولی از طرفی تکرار و اشتراک دارند و تنها المان غیر مشترکشان همان فلگ کمکی است پس می توانیم از بخش مشترک فاکتور بگیریم که می شود :

$$I\,T_2(m' + m): \quad NOP$$

که عبارت $(m' + m)$ برابر با ۱ میشود پس می توانیم به صورت کلی به جای ۲ خط در ۱ خط به شکل زیر بنویسیم :

$$I\,T_2: \quad NOP$$

همین کار مشابه را میتوانیم برای شرطی که I برابر بود با ۰ یعنی **آدرس غیر مستقیم** انجام دهیم، در ابتدا ۴ خط داریم :

چون معموری به رجیستر آدرس مستقیم متصل نیست باید از رجیستر دیتا به عنوان واسط استفاده کنیم

$$m'I'T_2: \quad DR \leftarrow M[AR]$$
$$m'I'T_3: \quad AR \leftarrow DR$$
$$m\,I'T_2: \quad DR \leftarrow M[AR]$$
$$m\,I'T_3: \quad AR \leftarrow DR$$

همانطور که ملاحظه می کنید ۲ اجرای یکسان داریم که هرکدام ۱ بار اضافه تکرار شده اند پس میتوانیم فرآیند فاکتور گیری را برای آنها دوباره انجام دهیم :

$$I'T_2\,(m' + m): \quad DR \leftarrow M[AR]$$
$$I'T_3(m' + m): \quad AR \leftarrow DR$$

که عبارت $(m' + m)$ برابر با ۱ میشود پس می توانیم به صورت کلی به شکل زیر بنویسیم :

$$I'T_2: \quad DR \leftarrow M[AR]$$
$$I'T_3: \quad AR \leftarrow DR$$

این مرحله در ۲ کلاک انجام شده است پس بدنه مشترک اجرای دستورات ما در مجموع ۴ کلاک میباشد.

Execute : $D_0D_1D_2 \dots D_{31}$

$$D_3(XNOR): \quad AC \leftarrow (AC \oplus M[AR])'$$

نکته ۲۴ : اجرای دستور یک نکته دارد، آن هم این است که ALSU های ما در یک کلاک میتواند فقط یک بار عملیات حسابی، منطقی، یا شیفت را انجام دهد ولی در این دستور مشاهده میکنیم که یک عملیات XOR ($\oplus$) وجود دارد و یک عملیات NOT که هرکدام در یک کلاک متفاوت باید اجرا شوند. حال چرا ؟ زیرا دستور XNOR را با علامت ($\odot$) نمایش می دهند و سوال می توانست از همین علامت استفاده کند ولی وقتی از XOR و NOT استفاده کرده یعنی ALSU ما دستور XNOR را ندارد و باید طراحی کنیم

نکته ۲۵ : جهت اجرای این دستور به این نکته باید توجه داشته باشیم که ابتدا بخش اول یعنی m' و سپس بخش دوم یعنی m را مینویسیم، همچنین اگر از بخش m' بخواهیم به بخش m برویم باید فلگ کمکی را یک کنیم و اگر بخواهیم از بخش دوم به بخش اول برویم باید فلگ کمکی را صفر کنیم، پیش از انجام این کار فرض کنیم که به جای دستور ۲ بخشی، ۱ بخش بیشتر نداریم پس ابتدا برای این حالت می نویسیم :

$$D_3T_4: \quad DR \leftarrow M[AR]$$
$$D_3T_5: \quad AC \leftarrow AC \oplus DR$$
$$D_3T_6: \quad AC \leftarrow AC' \ , \ SC \leftarrow 0$$

این RTL که نوشتیم برای فرمت دستور تک بخشی بود، اگر بخواهیم ۲ بخشی کنیم یعنی هم برای حالت m' بنویسیم و هم m، باید ابتدا **نکته ۲۲** را درنظر بگیریم، اگر رجیستر DR نصف خانه های حافظه که ۳۲ بیت بودند ظرفیت داشت، یعنی ۱۶ بیت، پس یکبار باید بخش اول را می ریختیم داخل DR و یکبار بخش دوم، یعنی :

$$m'D_3T_4: \quad DR \leftarrow M[AR]_{(15-0)}$$
$$m\,D_3T_4: \quad DR \leftarrow M[AR]_{(31-16)}$$

ولی چون رجیستر DR ما همان ۳۲ بیت است پس لازم نیست تفکیک کنیم و می توانیم کل یک خانه حافظه را درون رجیستر DR بریزیم :

$$D_3T_4: \quad DR \leftarrow M[AR]$$

خط دوم هم چون از موری چیزی خوانده نمی شود و تبادل بین رجیستر ها با ظرفیت مشترک ۳۲ بیت می باشد، خط دوم نیز همان میشود ولی برای خط سوم قسمت عملیاتی همان است منتها قسمت صفر کردن SC متفاوت می شود، چرا ؟ زیرا اگر ما در **بخش اول** باشیم و دستور بخش اول را انجام داده باشیم **باید m را یک کنیم** که برویم به بخش دوم، همچنین **SC کانتر را باید یک کنیم**، چرا یک ؟ زیرا اگر SC یک شوند زمان T_1 انتخاب میشود که یعنی مرحله Decode، ما مرحله Fetch را رد کردیم زیرا Fetch می آید و از حافظه می خواند ولی ما یک بار کل فرمت دستور را از حافظه خوانده ایم و مرحله اول بخش اول را دیکد کرده ایم حال کافی است بخش دوم را نیز در مرحله دوم دیکد کنیم. حال اگر در **بخش دوم** باشیم و دستور بخش دوم را انجام داده باشیم، **باید m را صفر کنیم** که برگردیم به بخش اول ولی حواستان باشد که باید دوباره Fetch کنیم زیر باید برگردیم به بخش اول دستور بعدی، پس کافیست **SC را صفر کنیم** که دوباره Fetch اتفاق بیفتد :

$$m'D_3T_6: \quad AC \leftarrow AC' \ , \ SC \leftarrow 1 \ , \ m \leftarrow 1$$
$$m\,D_3T_6: \quad AC \leftarrow AC' \ , \ SC \leftarrow 0 \ , \ m \leftarrow 0$$

همانطور که مشاهده میکنید، در این ۲ خط نیز بخش تکراری داریم که خارج از شرط انجام می شود، پس با خلاصه سازی و فاکتور گیری خواهیم داشت :

$$D_3T_6 \quad : \quad AC \leftarrow AC'$$
$$m'D_3T_6: \quad SC \leftarrow 1 \ , \ m \leftarrow 1$$
$$m\,D_3T_6: \quad SC \leftarrow 0 \ , \ m \leftarrow 0$$

ولی این کار باعث شد که به جای ۲ خط، ۳ خط بشود پس می توانیم برای این مرحله فاکتور گیری را انجام ندهیم و همان ۲ خط را بنویسیم. به صورت کلی داریم :

$$D_3T_4 \quad : \quad DR \leftarrow M[AR]$$
$$D_3T_5 \quad : \quad AC \leftarrow AC \oplus DR$$
$$m'D_3T_6: \quad AC \leftarrow AC' \ , \ SC \leftarrow 1 \ , \ m \leftarrow 1$$
$$m\,D_3T_6: \quad AC \leftarrow AC' \ , \ SC \leftarrow 0 \ , \ m \leftarrow 0$$

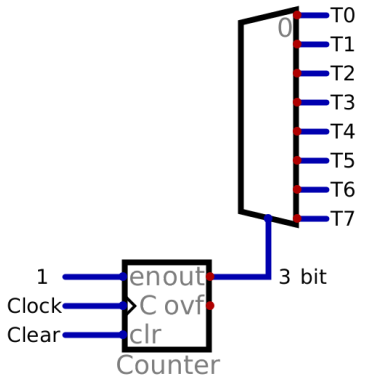
$$D_{12}(SUBM): \quad M[AR] \leftarrow AC - M[AR]$$

$$D_{12}T_4 \quad : \quad DR \leftarrow M[AR]$$
$$D_{12}T_5 \quad : \quad AC \leftarrow AC - DR$$
$$D_{12}T_6 \quad : \quad DR \leftarrow AC$$
$$m'D_{12}T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 1 \ , \ m \leftarrow 1$$
$$m\,D_{12}T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 0 \ , \ m \leftarrow 0$$

$$D_{15}(ADDM): \quad M[AR] \leftarrow AC + M[AR]$$

$$D_{15}T_4 \quad : \quad DR \leftarrow M[AR]$$
$$D_{15}T_5 \quad : \quad AC \leftarrow AC + DR$$
$$D_{15}T_6 \quad : \quad DR \leftarrow AC$$
$$m'D_{15}T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 1 \ , \ m \leftarrow 1$$
$$m\,D_{15}T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 0 \ , \ m \leftarrow 0$$

5. **رسم Sequencer :** پس از رسم RTL ها درمیابیم که بیشترین زمانی که طی شده تا دستور انجام شود T_7 است، حال این ۸ حالت زمانی را با چند عدد میتوانیم کد کنیم ؟ با ۳ بیت زیرا $۸ = ۲^۳$ پس داریم :



6. **پایه ها :** حال باید بررسی کنیم که پایه های مموری و رجیستر ها در چه زمان هایی فعال می شوند، به صورت کلی با بررسی DPU به پایه های زیر میرسیم :

$Read : T_0 + T_2 + D_3T_4 + D_{12}T_4 + D_{15}T_4$

$Write : D_{12}T_7 + D_{15}T_7$

$Clear(SC) : mD_3T_6 + mD_{15}T_7 + mD_{15}T_7$

$Clear(m) : mD_3T_6 + mD_{15}T_7 + mD_{15}T_7$

$Set(SC,\ 1) : m'D_3T_6 + m'D_{15}T_7 + m'D_{15}T_7$

$Set(m,\ 1) : m'D_3T_6 + m'D_{15}T_7 + m'D_{15}T_7$

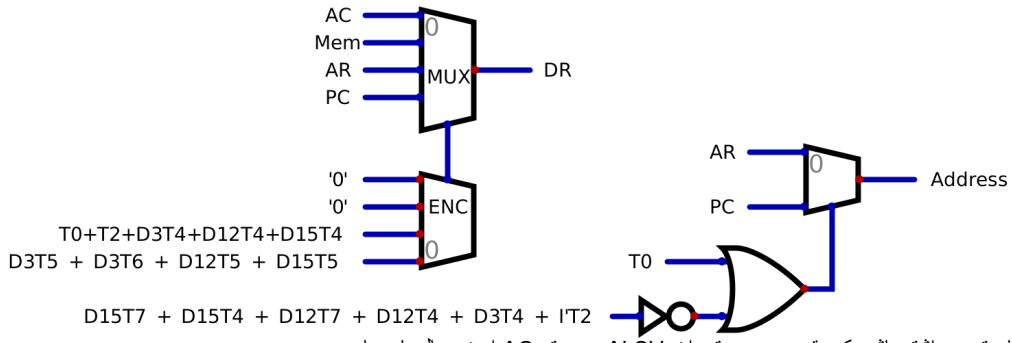
$LD(AR) : T_1 + I'T_3$

$LD(DR) : D_3T_4 + D_{12}T_4 + D_{12}T_6 + D_{15}T_4 + D_{15}T_6$

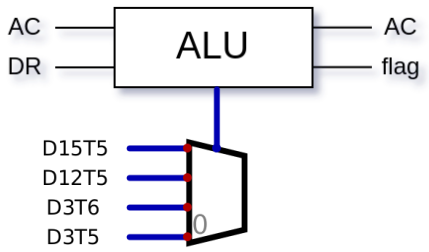
$LD(AC) : D_3T_5 + D_3T_6 + D_{12}T_5 + D_{15}T_5$

$INC(PC) : T_0$

7. **طراحی Bus :** در این DPU ما ۲ باس داریم که به یکی ۲ المان و به دیگری ۴ المان متصل است که به یکدیگر راه دارند و از آنجایی که باس را با مالتی پلکسر طراحی میکنند، میتوانیم یک مالتی پلکسر با ۲ بیت سلکت و یک انکودر با ۴ ورودی داشته باشیم و همچنین یک مالتی پلکسر با ۱ بیت سلکت بدون هیچ انکودری برای آن به شکل های زیر داشته باشیم :

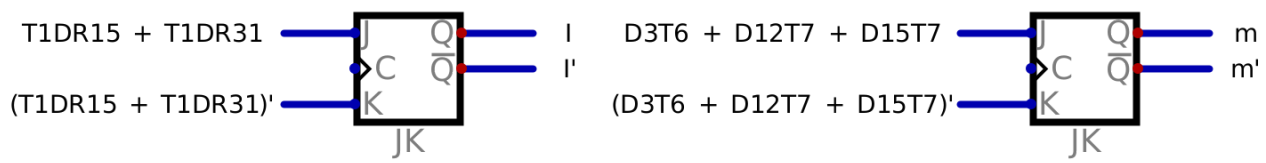


8. **دستورات ALSU :** باید توجه داشته باشیم که مقصد همه دستورات ALSU رجیستر AC است. حال باید بیاییم و در RTL های نوشته شده ببینیم که کجا از ALSU استفاده کرده ایم و سپس آن را طراحی کنیم :

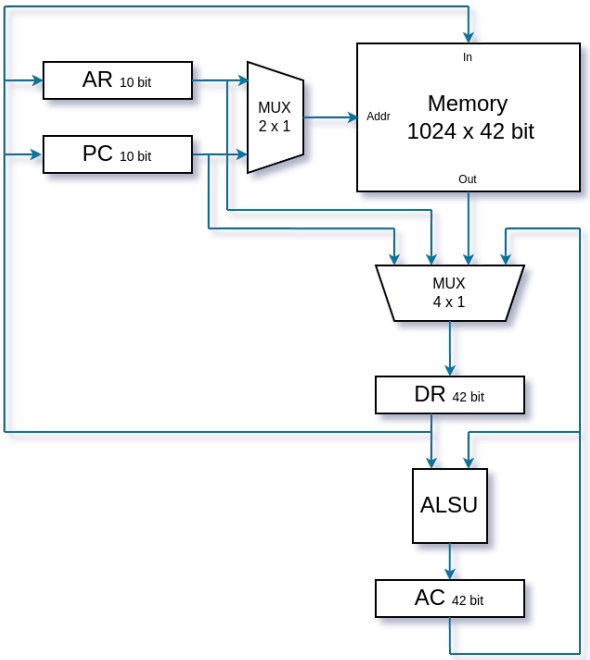


C_1	C_2	**ALU**
0	0	$AC \oplus DR$
0	1	$NOT(AC)$
1	0	$AC - DR$
1	1	$AC + DR$

9. **طراحی فلگ ها :** فلگ ها را با f . f JK (جی کا فلیپ فلاپ) طراحی میکنیم. در DPU داده شده یک فلگ I داریم و همچنین فلگ m را هم که خودمان اضافه کردیم پس کافیست برای این ۲ فلگ طراحی کنیم :



Question 9 - *Tir 1402, Dey 1401 Final Exam*



D_7	ADDM	111	$M[AR] \leftarrow AC + M[AR]$
D_3	SUB	011	$M[AR] \leftarrow AC - M[AR]$
D_5	BPNZ	101	$if\ (AC < 0)\ then\ PC \leftarrow AR$

I_1	$Opcode_1$	$Address_1$	I_2	$Opcode_2$	$Address_2$	I_3	$Opcode_3$	$Address_3$
1	3	10	1	3	10	1	3	10

Direct : I = 1
Indirect : I = 0

Solution :

ابتدا مراحل گفته شده را انجام میدهیم :

1. **عددگذاری مالتی پلکسر ها :** ۲ باس داریم، مانند سوال قبل عدد گذاری میکنیم. فرمت دستور نیز در مجموع ۴۲ بیت است که همان ۴۲ بیت از داخل حافظه بیرون می آید.
2. **رجیستر IR :** رجیستر IR در اینجا وجود ندارد پس باید ببینیم از کدام رجیستر میتوانیم به عنوان رجیستر دستور استفاده کنیم. با کمی دقت در میابیم که داده وقتی از ممووری خارج می شود مستقیم به رجیستر DR میرود پس میتوانیم از همان به عنوان رجیستر دستور استفاده کنیم.
3. **عددگذاری پیشوندهای Opcode ها :** همانطور که در جدول میبینید با توجه به ۳ بیت داده شده دستورات را از D_0 الی D_7 عدد گذاری کرده ایم که در اینجا فقط دستورات D_3، D_5 و D_7 استفاده شده اند.
4. **نوشتن RTL دستورات :** برای نوشتن RTL های دستورات باید تمامی مراحل چرخه دستورات را بنویسیم :

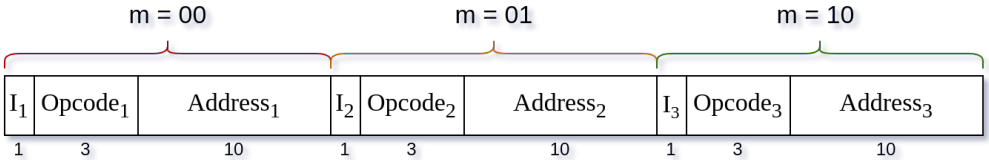
Fetch : $IR \leftarrow M[PC]\ ,\ PC \leftarrow PC + 1$

T_0 : $DR \leftarrow M[PC]\ ,\ PC \leftarrow PC + 1$ زیرا پروگرم کانتر مستقیم به ممووری وصل است

همانطور که می بینیم این مرحله فقط ۱ کلاک طول کشیده که انجام شود. همچنین توجه داشته باشید که از پایه INC برای اضافه کردن PC استفاده کرده ایم.

Decode : در این مرحله خانه حافظه ما ۳ دستور دارد پس در ۳ حالت باید دستور را دیکد کنیم

برای حل این بخش ما نیاز داریم که فرمت دستور را ۳ بخش کنیم که یکبار بخش اول، بار دیگر بخش دوم و در نهایت بخش سوم را پردازش کنیم، در نتیجه ۳ حالت داریم پس این ۳ حالت را با ۲ بیت میتوانیم کد کنیم. نام این بیت را m میذاریم و خواهیم داشت :



حال کافی است با توجه به ۳ حالت m، مراحلی که نیاز است این فلگ کمکی را تاثیر دهیم. وقتی میخواهیم دستور را دیکد کنیم، یک بار باید Opcode 1 و بار دیگر باید Opcode 2 و سپس Opcode 3 را دیکد کنیم پس باید برای اینکه مشخص کنیم آپکد کدام بخش را دیکد کنیم، چون فلگ کمکی ما ۲بیت است پس اندیس هایی به هر ۲ بیت آن اختصاص می دهیم (m_1 , m_2) در نتیجه :

$$m:\ 2\ bit\ \Rightarrow\ m = m_1 m_0$$

$m_1'm_0'T_1 : \ I \leftarrow DR_{(13)} \ , \ D_0D_1D_2 \dots D_{31} \leftarrow DR_{(12-10)} \ , \ AR \leftarrow DR_{(9-0)}$

$m_1'm_0 T_1 : \ I \leftarrow DR_{(27)} \ , \ D_0D_1D_2 \dots D_{31} \leftarrow DR_{(26-24)} \ , \ AR \leftarrow DR_{(23-14)}$

$m_1 m_0'T_1 : \ I \leftarrow DR_{(41)} \ , \ D_0D_1D_2 \dots D_{31} \leftarrow DR_{(40-38)} \ , \ AR \leftarrow DR_{(37-28)}$

این مرحله همانطور که میبینید با ۱ کلاک انجام شده است ولی به ۳ حالت متفاوت که فلگ کمکی مشخص میکند کدام خط اجرا شود.

Effective Address : $if \ (I = 0) \ then \ AR \leftarrow M[AR]$

با توجه به نکات مثال قبل داریم :

$I \, T_2 : \quad NOP$

$I'T_2 : \quad DR \leftarrow M[AR]$

$I'T_3 : \quad AR \leftarrow DR$

این مرحله در ۲ کلاک انجام شده است پس بدنه مشترک اجرای دستورات ما در مجموع ۴ کلاک میباشد.

Execute : $D_0D_1D_2 \dots D_{31}$

$D_7(ADDM) : \quad M[AR] \leftarrow AC + M[AR]$

با توجه به نکات مثال قبل و اینکه به ترتیب بخش به بخش باید بریم جلو و در آخرین بخش SC را صفر کنیم به صورت کلی داریم :

$D_7T_4 \quad : \quad DR \leftarrow M[AR]$

$D_7T_5 \quad : \quad AC \leftarrow AC + DR$

$D_7T_6 \quad : \quad DR \leftarrow AC$

$m_1'm_0'D_7T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 1 \ , \ m_0 \leftarrow 1$

$m_1'm_0 D_7T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 1 \ , \ m_1 \leftarrow 1 \ , \ m_0 \leftarrow 0$

$m_1 m_0'D_7T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 0 \ , \ m_1 \leftarrow 0$

چون عملیات تکراری داریم میتوانیم خلاصه بنویسیم ولی چون این ۳ خط تبدیل به ۴ خط میشوند نمی نویسیم.

$D_3(SUB) : \quad M[AR] \leftarrow AC - M[AR]$

$D_3T_4 \quad : \quad DR \leftarrow M[AR]$

$D_3T_5 \quad : \quad AC \leftarrow AC - DR$

$D_3T_6 \quad : \quad DR \leftarrow AC$

$m_1'm_0'D_3T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 1 \ , \ m_0 \leftarrow 1$

$m_1'm_0 D_3T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 1 \ , \ m_1 \leftarrow 1 \ , \ m_0 \leftarrow 0$

$m_1 m_0'D_3T_7 \quad : \quad M[AR] \leftarrow DR \ , \ SC \leftarrow 0 \ , \ m_1 \leftarrow 0$

$D_5(BPNZ) : \quad if \ (AC < 0) \ then \ PC \leftarrow AR$

$s \, D_5T_4 \quad : \quad DR \leftarrow AR$

$s \, D_5T_5 \quad : \quad PC \leftarrow DR$

$m_1'm_0'D_5T_5 \quad : \quad SC \leftarrow 1 \ , \ m_0 \leftarrow 1$

$m_1'm_0 D_5T_5 \quad : \quad SC \leftarrow 1 \ , \ m_1 \leftarrow 1 \ , \ m_0 \leftarrow 0$

$m_1 m_0'D_5T_5 \quad : \quad SC \leftarrow 0 \ , \ m_1 \leftarrow 0$

مراحل بعد را لطفا خودتان با توجه به نکات مثال های قبل بنویسید