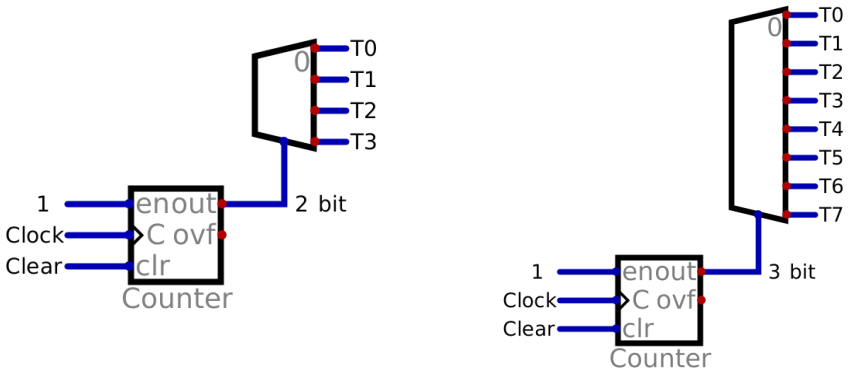
حل مسائل فصل ۵

کلاس حل تمرین معماری کامپیوتر دکتر زینالی، رامتین کوثری، ۲۹ آبان ۱۴۰۳ کلاس ۲۰۳

چرخه اجرای دستورات :

برای اینکه دستورات اجرا شوند چندین مرحله باید اتفاق بیفتد که ما در اینجا ۴ مرحله Effective Address، Decode، Fetch و Execute را بررسی میکنیم. توجه داشته باشید که دستورات در کلاک ها و زمان های مختلفی اجرا میشوند که با T_n نمایش میدهیم. این زمان ها را ماژولی به نام Sequence Counter برای ما فعال میکند که میتواند n بیتی باشد :



توجه داشته باشید که Sequencer را پس از نوشتن RTL های چرخه دستورات مینویسیم زیرا در ابتدا نمی دانیم که چند متغیر زمانی داریم و با نوشتن RTL ها به زمان ها پی می بریم. در Sequencer های بالا پایه های Enable همیشه ۱ هستند، پایه های Clock همیشه متصل به کلاک سیستم هستند و پایه Clear ما پایه ای است که Sequencer را ریست میکند. در Sequencer پیچیده تری که بعدا طراحی میکنیم، پایه SC هم داریم که به ما اجازه میدهد Sequencer را به شمارنده دلخواه برگردانیم که در حافظه هایی که هر خانه شان از ۲ یا چند دستور تشکیل شده کاربرد دارد و کاربرد دیگر آن این است که به جای استفاده از پایه Clear، پایه SC را صفر میکنند.

1. **مرحله اول Fetch :** دستورات ما در خانه های حافظه قرار دارند، ممکن است در یک خانه حافظه یک یا چند دستور موجود باشد که ما باید آنها را به ترتیب اجرا کنیم. برای اجرای دستورات ابتدا باید آنها را از حافظه بخوانیم (Fetch کردن) زیرا دستور های معماری M.R یا Memory Reference در حافظه قرار دارند، توجه داشته باشید که **در این مرحله ما فقط و فقط ۱ خانه از حافظه را میخوانیم** پس ممکن است در این ۱ خانه چندین دستور وجود داشته باشد که باید به ترتیب پردازش شوند. این دستورات خوانده شده از حافظه باید در یک رجیستر نگهداری شود که معمولا رجیستر IR میباشد ولی ممکن است این رجیستر در DPU مسئله موجود نباشد که زمانی که به این شکل بود باید ببینیم خروجی مموری به کدام رجیستر می رود و آن رجیستر را به عنوان رجیستر ذخیره دستور در کلاک مربوطه استفاده کنیم، با این کار رجیستر ما علاوه بر وظیفه قبلی خود وظیفه نگهداری دستورات را هم به عهده میگیرد. RTL مرحله Fetch دستور به شکل زیر است :

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

همانطور که در RTL مشخص است ما باید به آدرس PC در حافظه مراجعه کنیم تا دستور را بخوانیم که در اکثر مواقع PC به صورت مستقیم به پایه آدرس حافظه متصل نیست پس باید PC را به AR ترنسفر کنیم تا مقدار AR مقدار PC شود و بتوانیم مرحله Fetch را انجام دهیم.

2. **مرحله دوم Decode :** در این مرحله ما باید دستوراتی که در رجیستر دستور ما ذخیره شده است را رمزگشایی کنیم و ۳ بخش هر دستور آن را جدا کنیم. فرمت دستورات ما به این شکل بود :

I	$Opcode$	$Address$

که I بیانگر مستقیم یا غیر مستقیم بودن آدرس حافظه، $Opcode$ بیانگر عملگر که اگر n بیت باشد یعنی میتوانیم 2^n عملگر داشته باشیم و $Address$ بیانگر آدرس دستور میباشد. اگر برای مثال ۳ بیت $Opcode$ داشته باشیم، میتوانیم ۸ دستور را به شکل $D_0 D_1 D_2 \dots D_7$ داشته باشیم و باید آنها را به مقدار $Opcode$ هر دستور نسبت دهیم یعنی اگر $Opcode$ برابر بود با عدد 110 یعنی باید از $D_{6\ (Decimal)} = D_{110\ (Binary)}$ استفاده کنیم. حال اگر در نظر بگیریم که دستور ما ۱۶ بیتی است و ۱ بیت فلگ، ۳ بیت **Opcode** و ۱۲ بیت **Address** داشته باشیم، **RTL** مرحله **Decode** دستور به شکل زیر است :

$$Decode: I \leftarrow IR_{(15)}\ ,\ D_0 D_1 D_2 \dots D_7 \leftarrow IR_{(14-12)}\ ,\ AR \leftarrow IR_{(11-0)}$$

ولی اگر رجیستر **IR** نداشتیم و مجبور بودیم دستور را در یک رجیستر دیگر مانند **DR** واکشی یا **Fetch** میکردیم :

$$Decode: I \leftarrow DR_{(15)}\ ,\ D_0 D_1 D_2 \dots D_7 \leftarrow DR_{(14-12)}\ ,\ AR \leftarrow DR_{(11-0)}$$

3. **مرحله سوم Effective Address :** در این مرحله ما با توجه به فلگ I باید بررسی کنیم که آیا آدرس مستقیم است یا خیر که با توجه به آن به صورت بازگشتی در حافظه به آدرس و دستور های دیگر برگردیم تا زمانی که فلگ I آن دستور مستقیم باشد و دیگر نیاز نباشد که آدرس غیر مستقیم را پردازش کنیم.

بررسی Flag مستقیم یا غیر مستقیم بودن حافظه :

اگر شکل روبرو را به عنوان حافظه در نظر بگیریم، هر خانه این حافظه تنها یک دستور در آن وجود دارد. اعداد سمت چپ آدرس خانه حافظه هستند. در این حافظه اگر ما دستور در خانه ۳۰۰ را **Fetch** کنیم، در مرحله **Decode** میبینیم که فلگ I ۱ است یعنی آدرس غیر مستقیم است و باید به سراغ بخش **Address** رفته و در آنجا ۲۰۰ را میبینیم پس باید به خانه ۲۰۰ برویم و دوباره دستوری که در خانه ۲۰۰ قرار دارد دارای آدرس حافظه غیر مستقیم است که در بخش **Address** آن ۱۰۰ نوشته شده پس به خانه ۱۰۰ جهش میکنیم و میبینیم که فلگ I آن برابر با صفر است یعنی آدرس مستقیم است. پس دیگر نیاز نیست کاری بکنیم.

به صورت کلی اگر آدرس غیر مستقیم باشد باید آدرس خانه ای که در بخش آدرس قرار دارد را در نظر بگیریم، در غیر این صورت هیچ کاری را انجام نمیدهیم.

اگر در نظر بگیریم که دستور ما ۱۶ بیتی است و ۱ بیت فلگ، ۳ بیت **Opcode** و ۱۲ بیت **Address** داشته باشیم، **RTL** مرحله **Decode** دستور به شکل زیر است :

	I	Opcode	Address
100	0	xxx	0
	. . .		
200	1	xxx	100
	. . .		
300	1	xxx	200

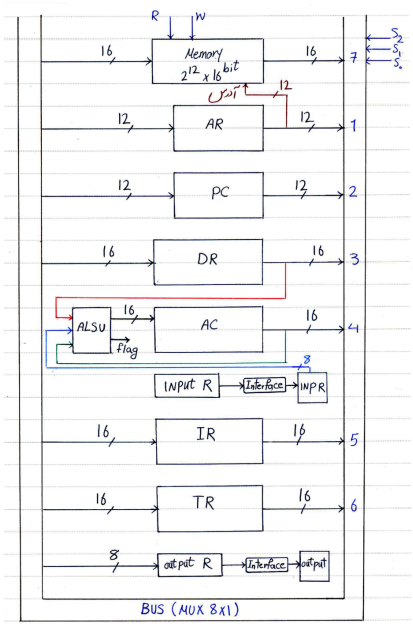
$$E.A:$$

$$I'T_n: NOP \quad (No\ Operation)$$

$$I\,T_n: AR \leftarrow M[AR]$$

توجه داشته باشید که به ۳ مرحله بالا بدنه مشترک گفته میشود. برای اجرای هر دستور حافظه، این ۳ مرحله ابتدا باید انجام شوند و سپس **Execute** صورت بگیرد منتها چون بدنه مشترک برای هر دستور **Opcode** تکراری است، ما فقط یکبار آن را در امتحان می نویسیم.

4. **مرحله چهارم Execute :** حال که بدنه مشترک دستور انجام شده، باید آن دستور را اجرا کنیم که اجرا کردن دستور بستگی به DPU دارد، برای مثال دستور و DPU زیر را داریم :



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

اگر بخواهیم زمان بندی و RTL این دستور را در مرحله Execute بنویسیم، باید DPU را بررسی کنیم که ببینم این RTL چگونه و در چند کلاک انجام میشود. پس از بررسی متوجه میشی که داده باید به DR برود و سپس از DR به AC به واسطه ALSU برود، پیش از نوشتن RTL فرض کنیم که بدنه مشترک در ۴ کلاک طول کشیده پس اجرای دستورات از T_4 شروع میشود، اگر در نظر بگیریم که دستور ما ۱۶ بیتی است و ۱ بیت فلگ، ۳ بیت Opcode و ۱۲ بیت Address داشته باشیم، RTL مرحله Decode دستور به شکل زیر است :

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

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

پس اجرای این دستور ۲ کلاک طول کشیده، ولی در ابتدا گفتیم که این Sequencer هستش که باعث فعال شدن T ها میشود اما حال که در زمان T_5 چرخه اجرای دستور ما تمام شده باید دوباره به اول بازگردیم پس عملا T_6 و T_7 استفاده نمیشوند پس میتوانیم Sequencer را Clear کنیم تا از صفر شروع شود یعنی $SC \leftarrow 0$:

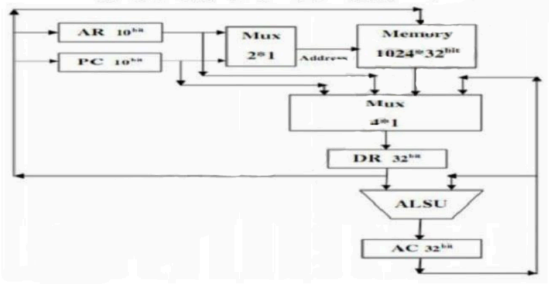
$$D_3T_5 : \quad AC \leftarrow DR\,,\, SC \leftarrow 0$$

مراحل حل سوال امتحان

در امتحان نمونه سوال ها به شکل روبرو طرح میشود. برای حل قسمت الف و ب این سوال باید مراحل زیر را طی کنیم :

1. مالتی پلکسر ها باس های ما هستند پس باید ورودی ها را عدد گذاری کنیم و سپس بیت های فرمت دستور را بنویسیم.
2. به دنبال رجیستر IR میگردیم که در DPU موجود نیست ولی در شکل میبینیم داده ای که از حافظه خوانده می شود مستقیم از ALSU رد شده و بدون استثناء به رجیستر DR می رود، پس می توانیم وظیفه نگهداری از دستور را نیز به همین رجیستر بدهیم پس تا اینجا DR هم برای AC استفاده می شود هم IR.
3. عدد گذاری پیشوند های Opcode ها
4. نوشتن RTL دستورات شامل مراحل چرخه اجرای دستورات
5. رسم Sequencer
6. نوشتن زمان های فعال شدن پایه ها
7. متصل کردن زمانبندی ها و المان ها به Bus های موجود
8. طراحی ALSU
9. طراحی فلگ ها

۴- واحد DPU(Data Path Unit) و نوع وتعدادی از دستورات یک پردازنده بصورت زیراست:



I_1	$Opcode_1$	$Address_1$	I_2	$Opcode_2$	$Address_2$
1^{bit}	5^{bit}	10^{bit}	1^{bit}	5^{bit}	10^{bit}

(I = 0 Indirect , 1 Direct)

Symbol	Opcode	Function
XNOR	00011	AC ← (AC ⊕M[AR])'
SUBM	01100	M[AR]← AC - M[AR]
ADDM	01111	M[AR] ← M[AR] + AC

الف) نحوه خواندن و اجرای کامل دستورات را بصورت RTL بنویسید. (نمره ۳)
ب) واحد کنترل را بصورت Hardwire بطور کامل طراحی کنید. (نمره ۳)
ج) ریز برنامه لازم برای اجرای کامل دستورات بصورت MicroProgram بنویسید. (نمره ۳)
د) فرمت ریز دستور مناسب برای MicroProgram فوق طراحی کنید. (نمره ۳)

موفق باشید.

از مرحله ۱ تا ۶ قسمت الف و مرحله ۷ تا ۹ قسمت ب سوال هستند. مثال های حل شده با توضیحات در پی دی اف سوالات قرار دارند.