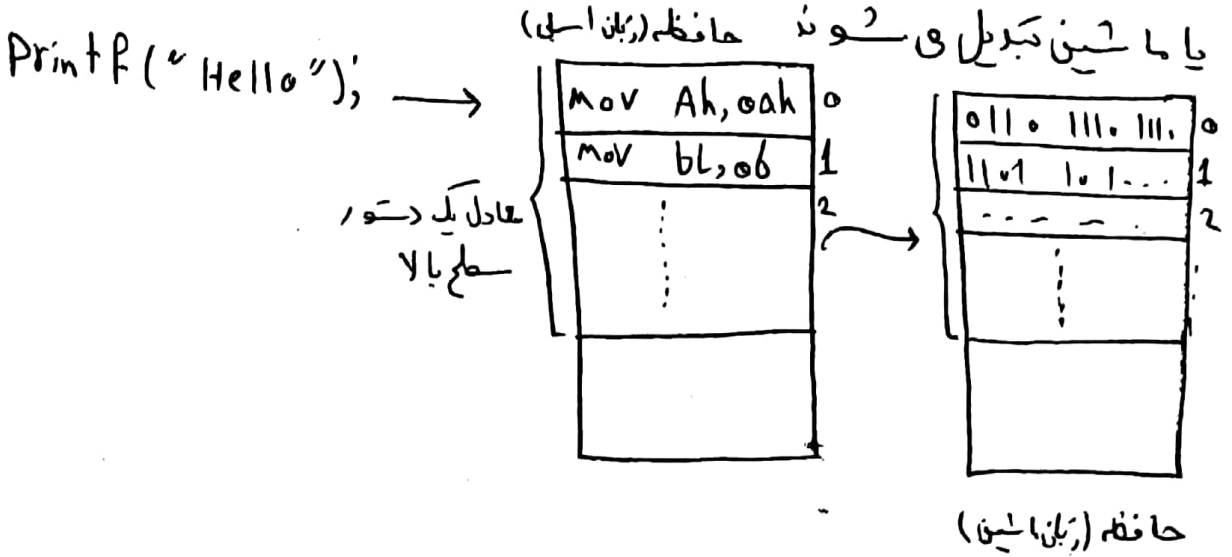
بنام خدا

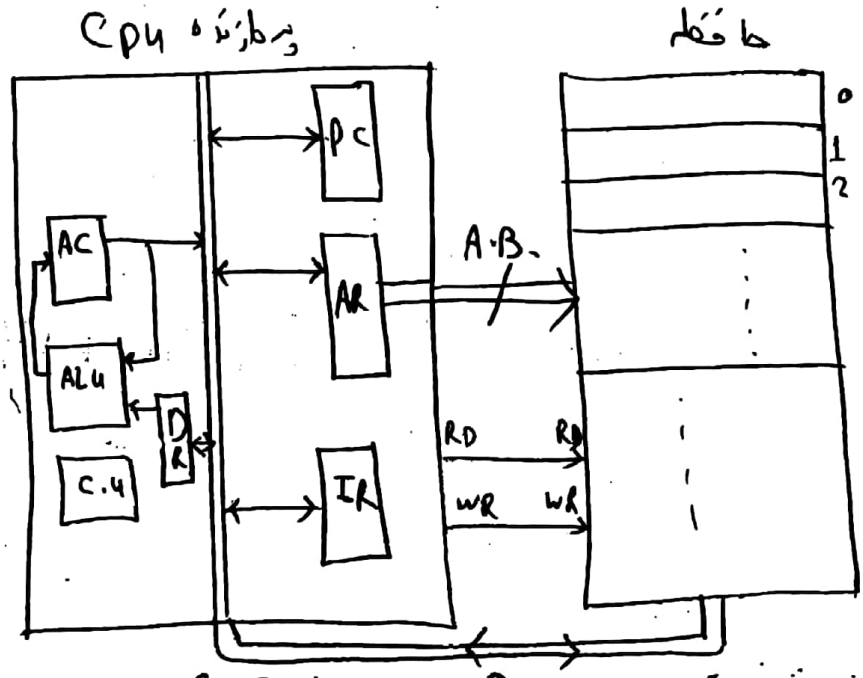
(1) معماری کامپیوتر
استاد: رکابی نژاد

طراحی یک کامپیوتر ساده:

- تمام دستورات زبان‌های سطح بالا برای اجرا، ابتدا به دستورات زبان اسمبلی یا ماشین تبدیل می‌شوند



- یک کامپیوتر برای اجرای این دستور بالا، زیر عمل می‌کند: (ماشین نیمن)



که هر دستور العمل، در ۳ فاز اجرا می‌شوند:

۱- واکشی
۲- تفسیر
۳- اجرا

CPU ابتدا دستورات زبان ماشین را یکی‌یکی واکشی نموده، سپس آن را تفسیر و در نهایت اجرا می‌کند. باید توجه داشت که برای اجرای یک دستور زبان ماشین، لازم است که چندین RTL اجرا

اهداف: طراحی یک پردازنده

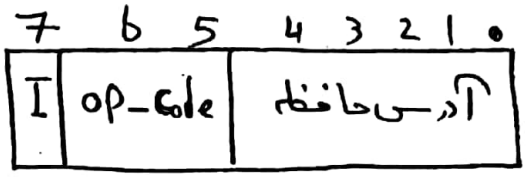
۱- در ابتدا باید مشخص کنیم که قالب دستورات زبان اسمبلی (ماشین) ما چگونه است

هر پردازنده دستورات مخصوص خود را دارد

MOV	R1, R2	یک دستورالعمل انتقال ۲ عملوند
ADD	R1, R2	~ ~ جمع ~ ~ ۲ ~
ADD	R1, R2, R3	⟵ MOV R1, M[10]
		R1 ← R2 + R3
MOV	R2	AC ← R2
MOV	M[10]	AC ← M[10]
ADD	M[7]	AC ← AC + M[7]

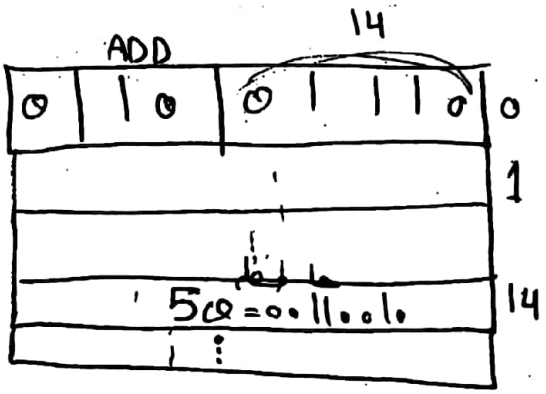
بسته به اینکه چه دستوراتی داریم، قالب دستورالعمل متفاوت خواهد بود.

در معماری کامپیوترهای ساده ما، قالب دستورالعمل ۸ بیتی است و دستورات تک عملوندی حافظه‌ای هستند



$I = \begin{cases} 0 & \text{شیوه آدرس‌دهی مستقیم} \\ 1 & \text{~ غیرمستقیم} \end{cases}$

ADD [14] ≡ باشیوه آدرس‌دهی مستقیم



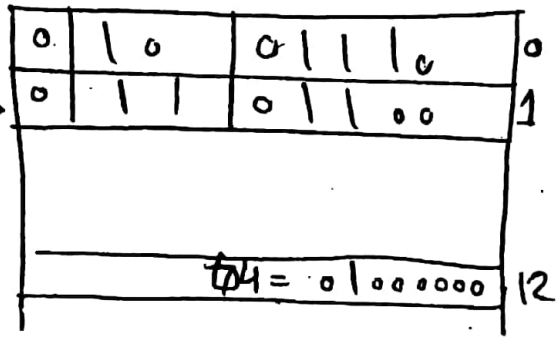
AC ← AC + M[14] معنی

AC ← AC + 50

~ ~ SUB [12]

AC ← AC − M[12]

AC ← AC − 64



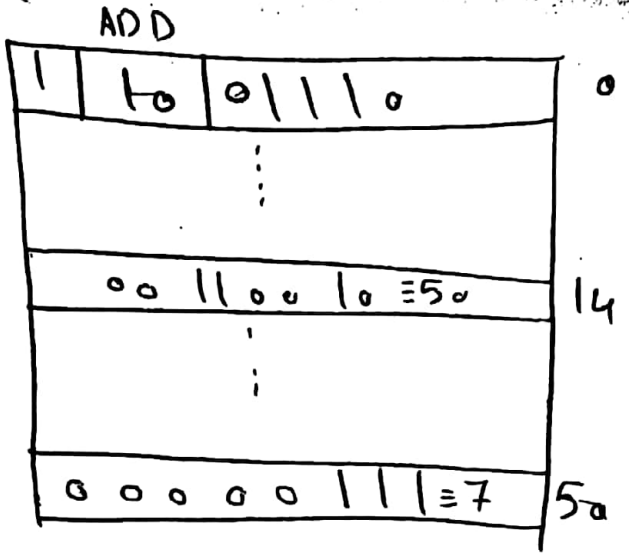
(۳)

غیرمستقیم ADD [14]

$AC \leftarrow AC + M[M[14]]$

$AC \leftarrow AC + M[50]$

$AC \leftarrow AC + 7$

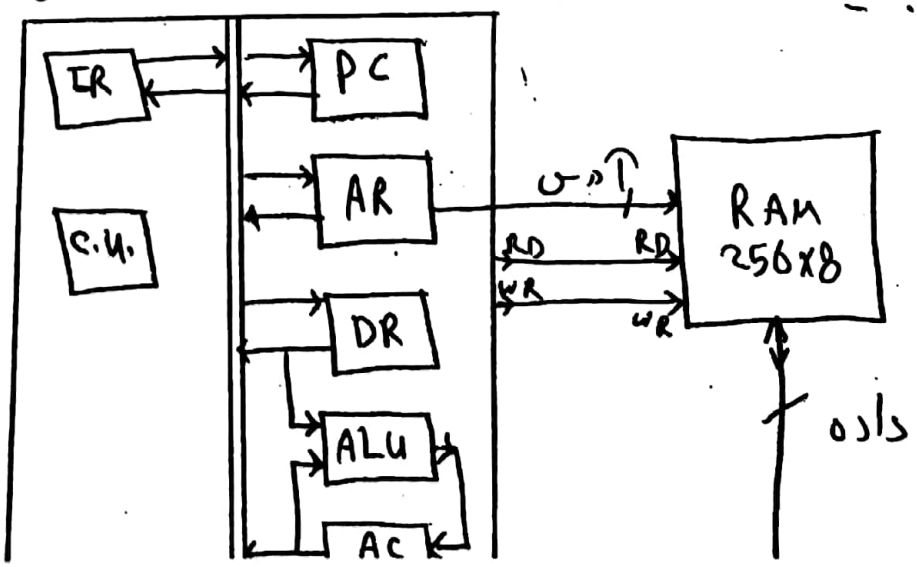


۲- مشخص نمودن معماری داخل cpu :

۴ نوع ثبات هائی داریم. وظیفه هرکدام چیست. نحوه اتصال این ثبات ها در باس مشترک چگونه است

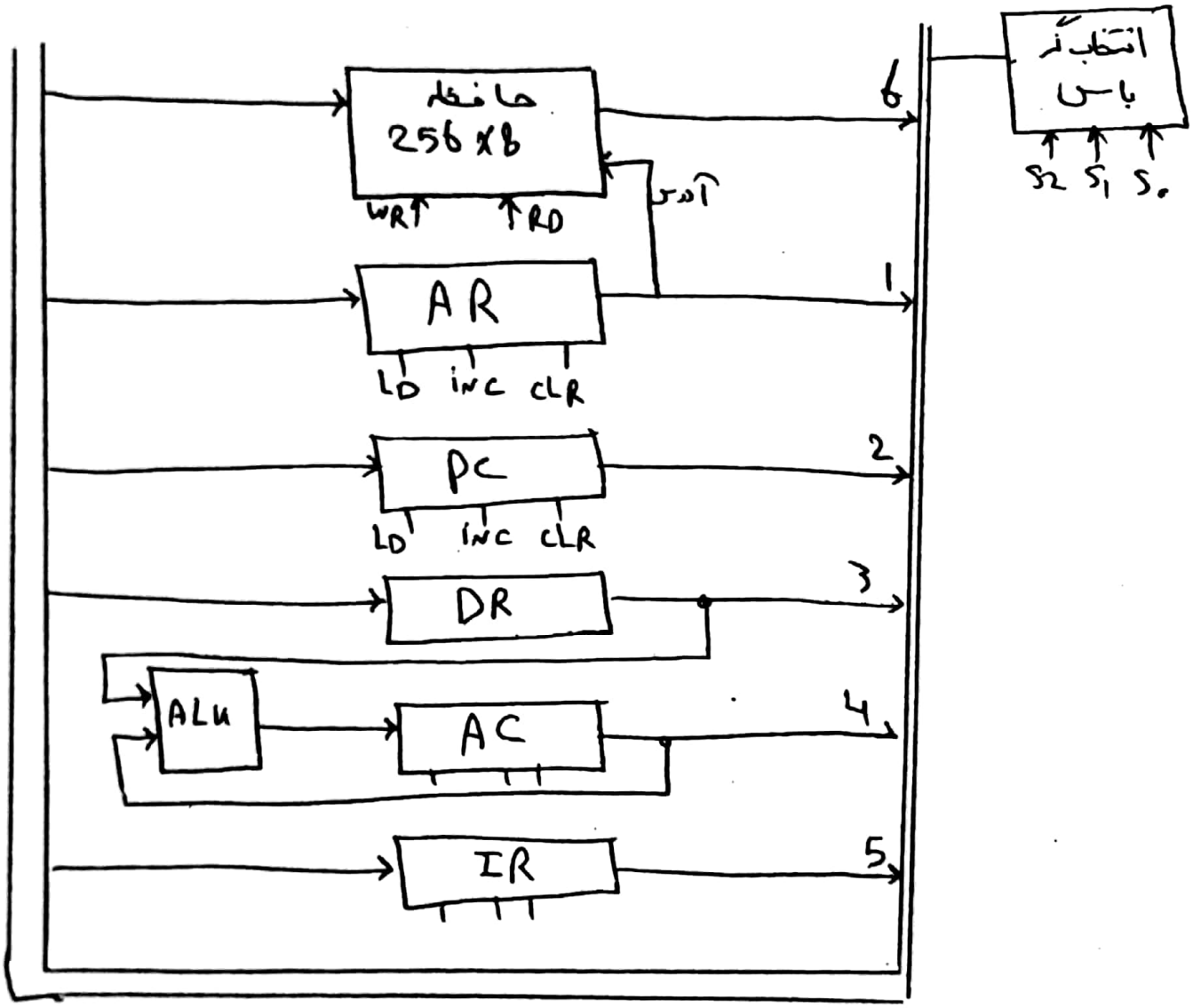
ثبات	تعداد بیت	وظیفه
AC	۸	ثبات انباشتگر : نتیجه محاسبات در این ثبات قرار می‌گیرد
IR	۸	ثبات دستورالعمل : دستورالعمل ها بعد از واکشی از حافظه در این ثبات قرار می‌گیرند
DR	۸	ثبات داده : داده ها ~ ~ ~ ~ ~ ~
AR	۸	ثبات آدرس : برای حافظه مشخص می‌کند که با کدام خانه از حافظه کار داریم
PC	۸	ثبات شمارنده برنامه : وظیفه آن نگهداری آدرس دستورالعمل بعدی است که باید اجرا گردد.

- حالا باید نحوه اتصال این ثبات ها درون cpu و باس مشترک آن مشخص شود



۴

- نمایی دیگر از اتصال ثبات ها و حافظه در یک باس مشترک :



انتخاب گر باس انتخاب می کند کدام یک از مقدار خود را بروی باس مشترک قرار دهند

۳- تعریف انواع دستورالعمل ها برای کامپیوتر با توجه به قالب دستورالعمل:

- انواع دستورالعمل های قابل تعریف:
  ۱- ریاضی
  ۲- منطقی
  ۳- جابجایی
  ۴- ورودی خروجی

ADD M[10] ← مفهوم ← $AC \leftarrow AC + M[10]$

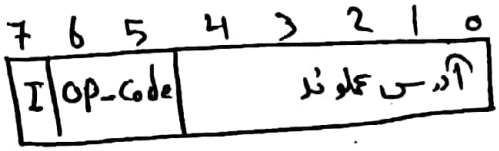
AND M[10] ← ~ ← $AC \leftarrow AC \wedge M[10]$

ShL M[10] ← ~ ← $AC \leftarrow shl\, M[10]$

IN 10 ← ~ ← $AC \leftarrow I/O[10]$

- دستورات از نظر عملوند
  - باقی
    - CLR A ← $AC \leftarrow 0$
    - CMA A ← $AC \leftarrow \overline{AC}$
  - حافظه ای
    - ADD M[10] ← $AC \leftarrow AC + M[10]$

با توجه به کم بودن تعداد بیت های op-Code (۲ بیت) فقط ۴ تا از دستورالعمل تعریف کنیم و عملوند ما هم فقط از نوع حافظه ای است.



op-Code 6 5	نوع دستور
0 0	Store
0 1	Load
1 0	Add
1 1	sub

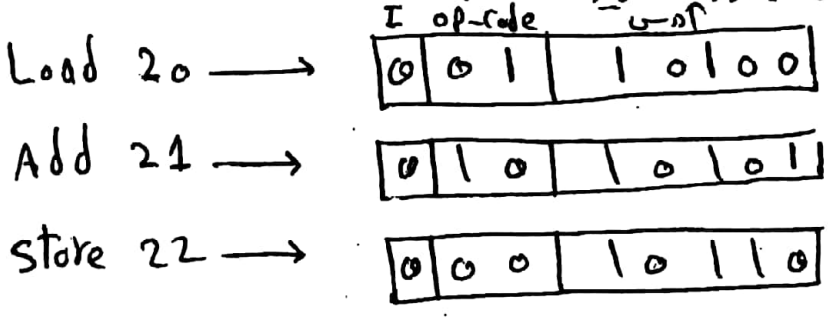
دستورات زیر را در نظر بگیرید:

Load 20 → مفهوم (محتوای خانه 20 حافظه را در AC قرار بده) $AC \leftarrow M[20]$

Add 21 → $AC \leftarrow AC + M[21]$

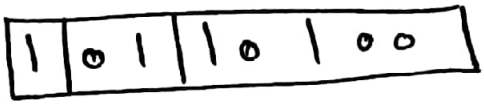
Store 22 → $M[22] \leftarrow AC$

معادل دستورات زبان ماشین فوق بصورت زیر است:



(9)

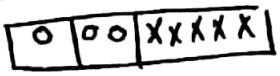
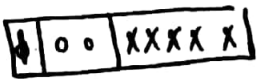
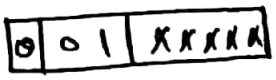
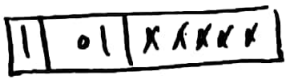
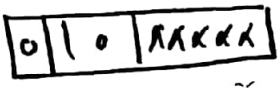
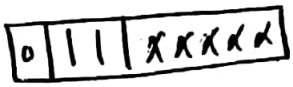
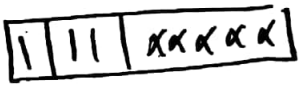
برای نمونه آدرس دهی غیرمستقیم :

Load 20 ⟶ 

مفهوم : $AC \leftarrow M[M[20]]$

⋮

در نهایت 8 دستورالعمل خواهیم داشت :

دستور	معادل زبان ماشین I=0	معادل زبان ماشین I=1	نوع عملیات
store			ذخیره AC در خانهٔ حافظه
load			ذخیره خانهٔ حافظه در AC
add			جمع کردن خانهٔ حافظه با AC و قرار دادن نتیجه در AC
sub			تفریق ~ ~ ~ ~ ~ ~ ~ ~

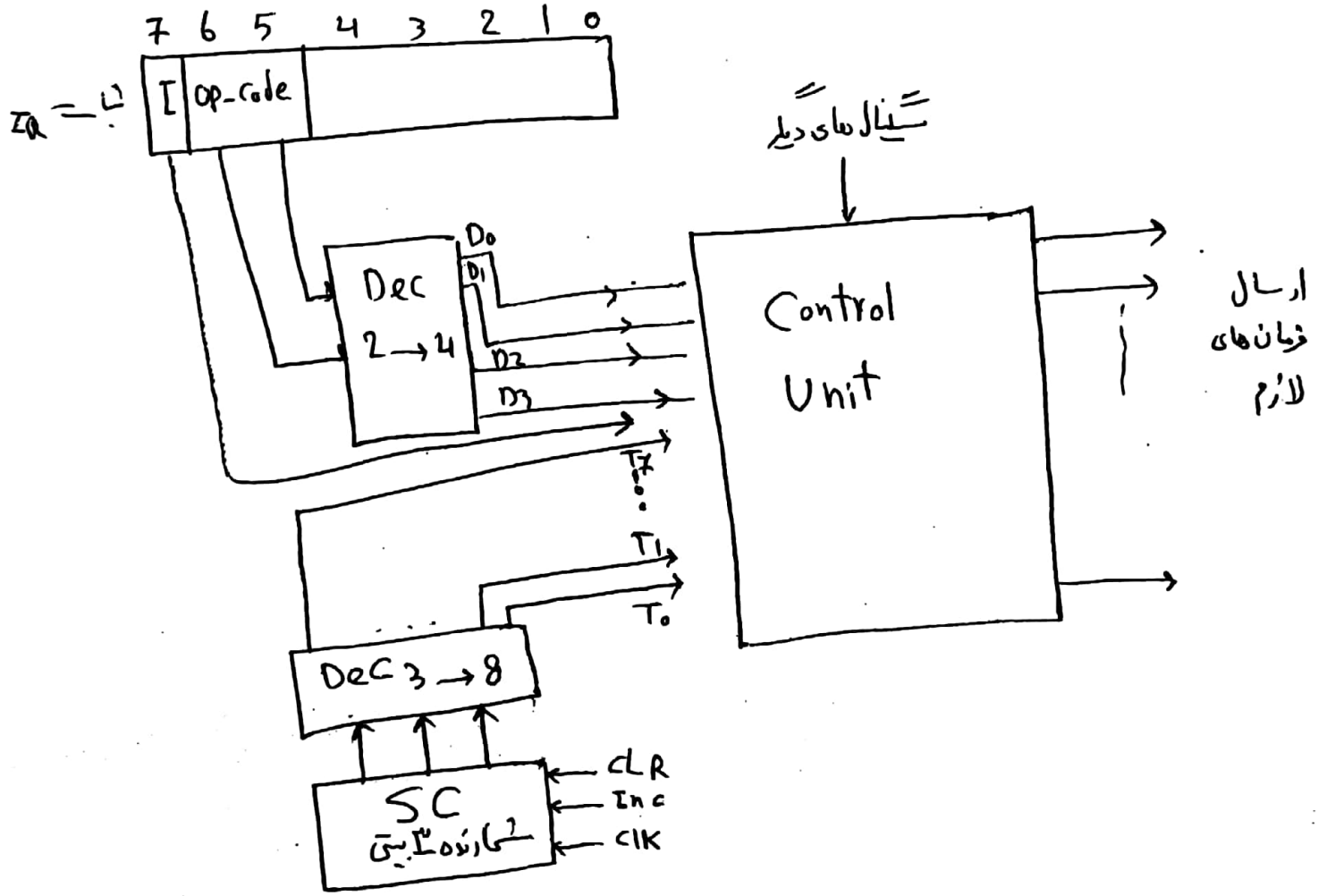
۴- طراحی C.U.:

وظایف C.U. عبارتند از:

۱- ارسال سیگنال های لازم برای دریافت دستورالعمل از حافظه به داخل CPU

۲- بررسی و درک دستورالعمل

۳- ارسال سیگنال های لازم برای اجرای دستورالعمل



همانطور که مثلاً گفته شد یک شمارنده برای اجرای یک دستور زبان ماشین، باید چندین ریزدستورالعمل را اجرا کند. مثلاً برای اجرای دستور LD 20 ، ریزعمل های زیری باید اجرا گردد:

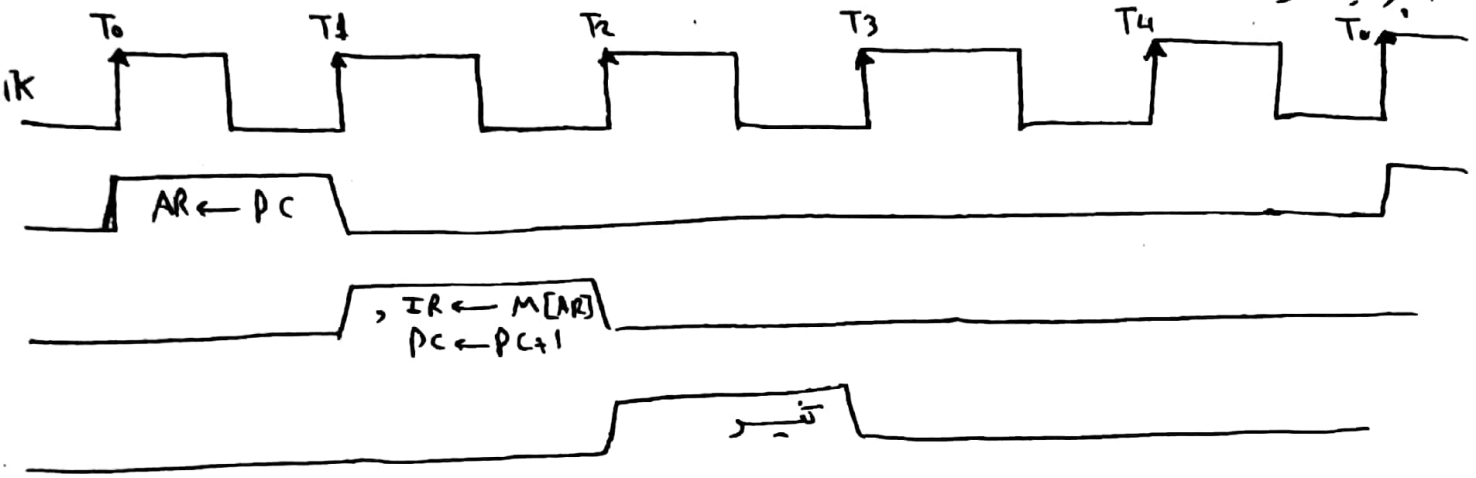
واکه | T_0 : $AR \leftarrow PC$

T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

T_2 : $D_3D_2D_1D_0 \leftarrow Decode(IR6, IR5)$, $AR(4-0) \leftarrow IR(4-0)$, $AR(7-5) \leftarrow 0$, $I \leftarrow IR7$

$T_3 \cdot I$: $AR \leftarrow M[AR]$

C.U. ی باید در هر یک از فاصله های زمانی T_0، T_1، ... (که میکروام همزمان خال می شوند)، سیگنال های لازم را برای اجرای هر یک از ریز دستورالعمل ها صادر نموده، تا نهایتاً یک دستورالعمل ماشین اجرا گردد.



- مراحل اجرای دستورالعمل توسط C.U. :

فرض در حافظه کامپیوتر، مجموعه‌ای از دستورالعمل ها وجود دارد. در کامپیوتر برای اجرای هر دستورالعمل نیاز است که چرخه زیر انجام پذیرد:

۱- واکشی یک دستورالعمل از حافظه به داخل CPU (fetch)

۲- رمزگشایی یا دیکد دستورالعمل (Decode)

۳- خواندن آدرس عوثر از حافظه، در صورتیکه دستورالعمل، آدرس غیرمستقیم داشته

۴- اجرای دستورالعمل (Execute)

(۹) -

۱- فاز واکشی (fetch)

در این فاز ۱ بایت دستورالعمل مورد نظر از حافظه به داخل CPU وارد می‌شود. آدرس دستورالعملی که باید از حافظه خوانده شود، در بایتی بنام PC (Program Counter) قرار دارد. این بایت همواره به دستورالعمل بعدی که باید اجرا شود اشاره دارد. وقتی اطلاعات از حافظه خوانده شد در بایتی بنام IR (Instruction Register) قرار می‌گیرد. می‌داریم:

T_0 : $AR \leftarrow PC$

T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

از آنجا که فقط بایت AR به پایه‌های آدرس حافظه متصل است، می‌باید ابتدا در فاز T_0، محتوای بایت PC را به AR وارد کنیم. سپس در فاز T_1، از حافظه به آدرس AR، اطلاعات را خوانده و وارد بایت IR می‌شود ($IR \leftarrow M[AR]$). چون قرار است بایت PC، همچنین به دستورالعمل بعدی که قرار است اجرا شود اشاره کند در فاز T_1، PC با یکواحد افزایش می‌یابد ($PC \leftarrow PC+1$).

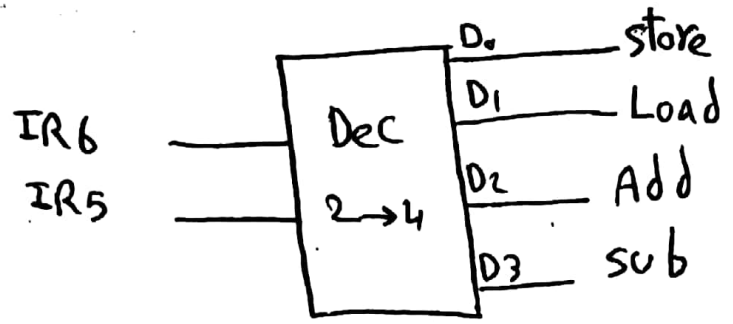
۲- فاز رمزگشائی (Decode)

در این فاز بایت O.C. درک کند که دستورالعمل واکشی شده، چه عملیاتی دارد

T_2 : $D_3 D_2 D_1 D_0 \leftarrow Decode(IR(6-5))$, $AR(4-0) \leftarrow IR(4-0)$,
$AR(7-6-5) \leftarrow 0$, $I \leftarrow IR(7)$

در این فاز ابتدا کامپیوتر بیت‌های پنجم و ششم بایت IR را یک دیکودر ۲ به ۴ وصل می‌شود تا براساس فعال بودن خروجی‌های D_0 تا D_3 درک کنیم کدام دستور

(10)



همچنین بیت هفتم ثبات IR را در F.F. بنام I قرار می‌دهد تا با بررسی آن مشخص شود که نوع آدرس دهی مستقیم است یا غیر مستقیم.

همچنین بیت‌های مربوط به آدرس در ثبات IR را در ثبات AR قرار می‌دهد

حال برای این ۳ تا پیغام زمانی باید دید که چه سیگنال‌هایی توسط C.U. باید فعال شوند تا عملیات بدرستی انجام پذیرد.

T_0 : $AR \leftarrow PC$

برای انجام این ریزعمل، C.U باید کنترل باس، ثبات PC را بروی باس قرار دهد.
پس $S_2S_1S_0 = 010$ باشد تا اطلاعات ثبات PC بروی باس مشترک قرار گیرد. حال C.U باید LD ثبات AR فعال شود تا این اطلاعات وارد ثبات AR شود.

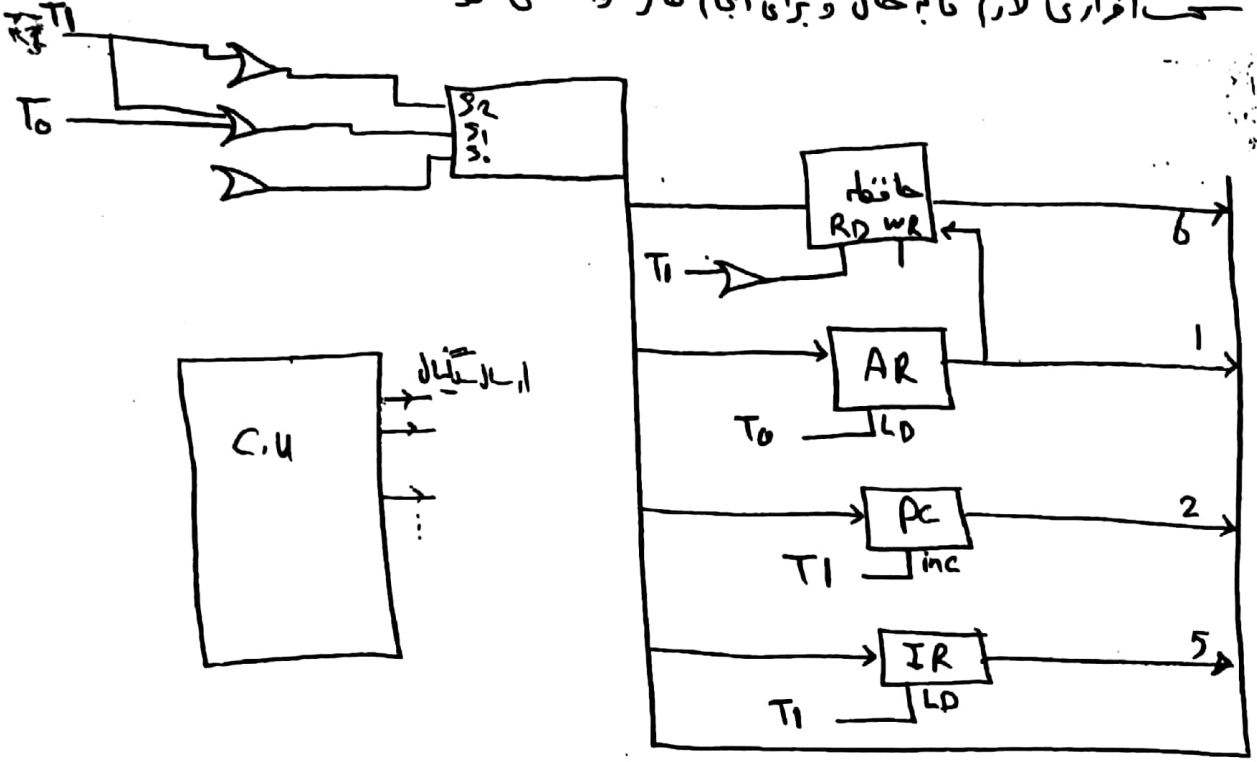
T1: $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

برای انجام $IR \leftarrow M[AR]$، ابتدا باید RD توسط C.U فعال شود تا با فعال شدن این پایه حافظه بعنوان عمل مورد نظر خواندن از حافظه است. پس $S_2S_1S_0 = 110$ باشد تا اطلاعات خوانده شده از حافظه بروی BUS مشترک قرار گیرد. همچنین LD ثبات IR نیز باید فعال شود تا این اطلاعات وارد ثبات IR شود. پایه inc ثبات PC نیز فعال شود تا محتویات یکواحد افزایش گردد.

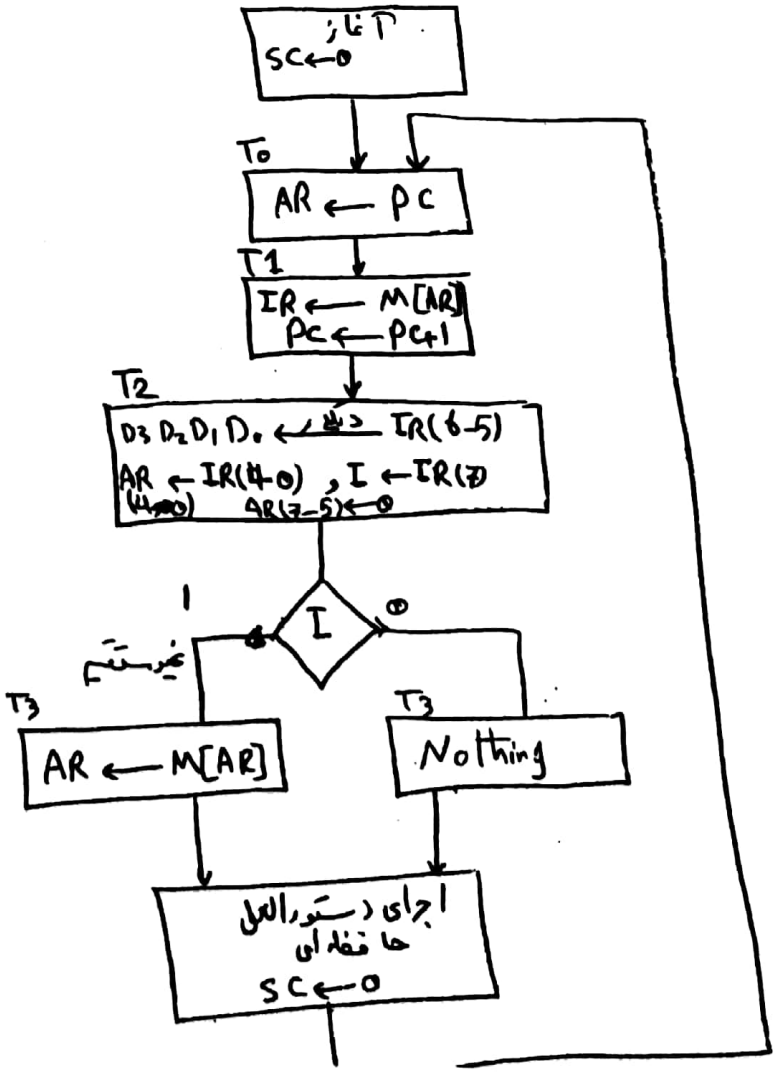
T2:

LD (AR) و $S_2S_1S_0 = 101$.

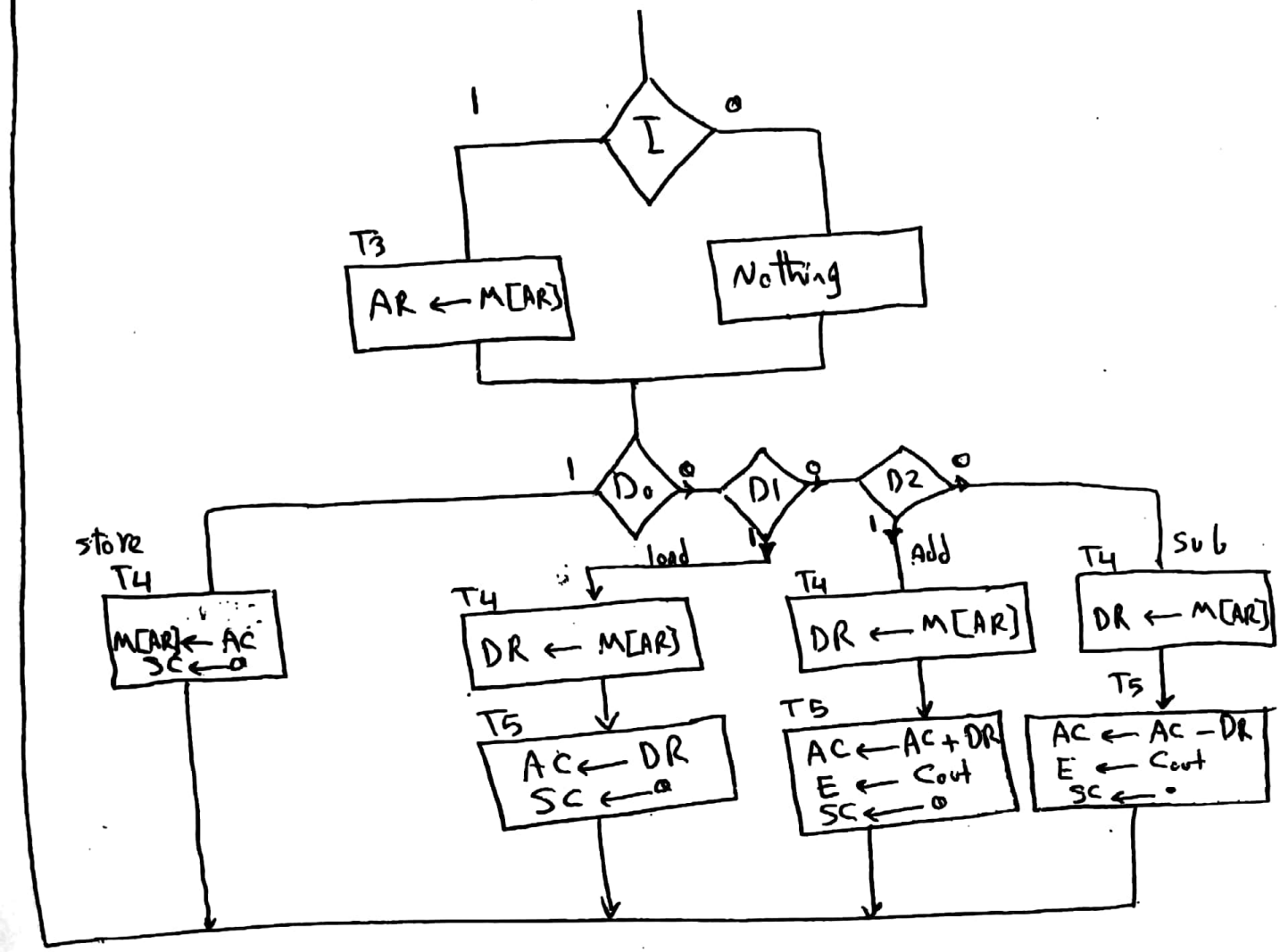
اتصالات سخت افزاری لازم کامپیوتر و برای انجام فاز واکشی توسط C.U. بصورت زیر است



حال برای انجام فاز آخر، یعنی فاز اجرا، CU فقط باید نوع دستورالعمل را تشخیص داده و سیگنال های لازم جهت اجرا را ارسال نماید.



فلوچارت مقابل برای فاز اجرا به صورت زیر کامل می شود



RTL های مربوط به فاز واکشی و دیکد برای همه دستورات یکسان است ولی فازهای اجرای آنها با هم فرق می کند:

T_0: $AR \leftarrow PC$

T_1: $IR \leftarrow M[AR], PC \leftarrow PC+1$

T_2: $D_3 \cdots D_0 \leftarrow Decode(IR(6-5)), AR(4-0) \leftarrow IR(4-0), AR(7-5) \leftarrow 0, I \leftarrow IR(7)$

IT_3: $AR \leftarrow M[AR]$

دستور store

$D_0 T_4$: $M[AR] \leftarrow AC, SC \leftarrow 0$

دستور Add

$D_2 T_4$: $DR \leftarrow M[AR]$

$D_2 T_5$: $AC \leftarrow AC + DR, E \leftarrow Cout, SC \leftarrow 0$

دستور load

$D_1 T_4$: $DR \leftarrow M[AR]$

$D_1 T_5$: $AC \leftarrow DR, SC \leftarrow 0$

دستور Sub

$D_3 T_4$: $DR \leftarrow M[AR]$

$D_3 T_5$: $AC \leftarrow AC - DR, E \leftarrow Cout, SC \leftarrow 0$

حال که تمام RTL های مربوط به این دستورالعمل را داریم می‌توانیم اتصالات سخت‌افزاری را کامل کنیم

- کنترل ورودی (LD) ثبات‌ها:

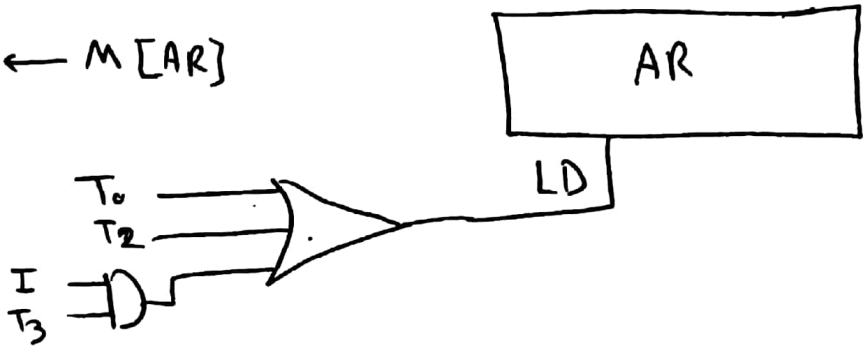
باید ببینیم کدام ریزعمل‌های توانند به ثبات موردنظر مقدار دهند

مثلاً برای ثبات AR:

T_0 : $AR \leftarrow PC$

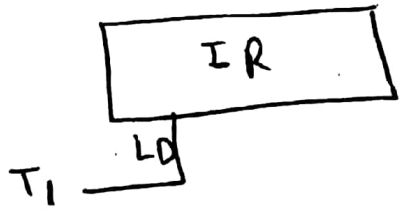
T_2 : $AR(4-0) \leftarrow IR(4-0)$

IT_3 : $AR \leftarrow M[AR]$



برای ثبات IR:

T_1 : $IR \leftarrow M[AR]$

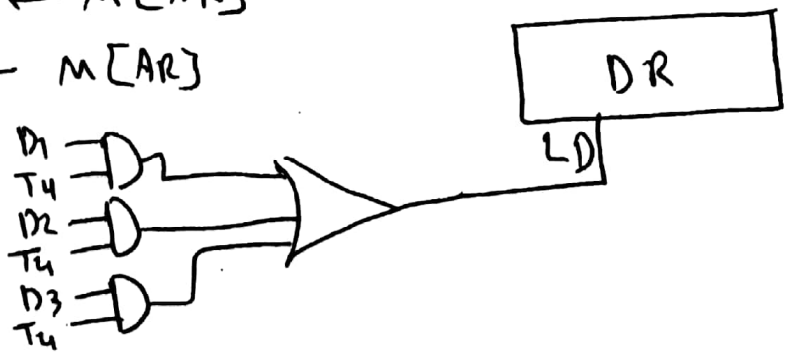


برای ثبات DR:

D_1T_4 : $DR \leftarrow M[AR]$

D_3T_4 : $DR \leftarrow M[AR]$

D_2T_4 : $DR \leftarrow M[AR]$

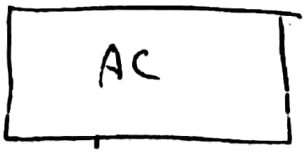


برای ثبات AC:

D_1T_5 : $AC \leftarrow DR$

D_2T_5 : $AC \leftarrow AC + DR$

D_3T_5 : $AC \leftarrow AC - DR$



- برای حافظه:

اگر بخواهیم مقداری را در حافظه وارد کنیم از $M[AR] \leftarrow DR$ استفاده می‌کنیم و آن را با WR فعال می‌کنیم

اگر بخواهیم مقداری را از حافظه بخوانیم مثلاً $DR \leftarrow M[AR]$ ، ~ ~ RD ~ ~ ~

- برای خواندن از حافظه:

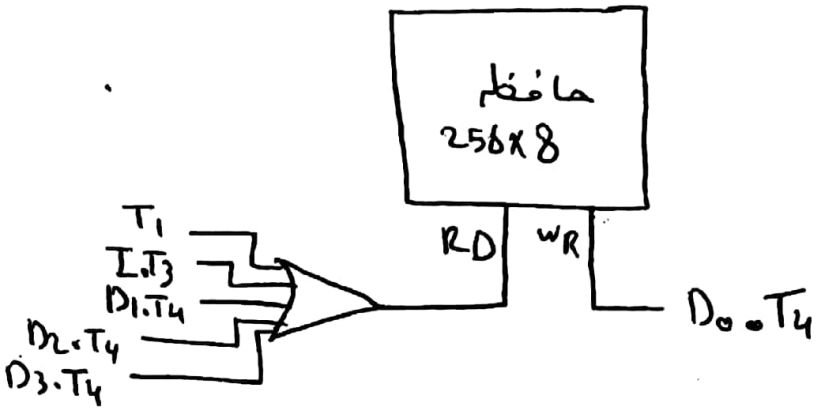
$T_1 : IR \leftarrow M[AR]$

$IT_3 : AR \leftarrow M[AR]$

$D_1T_4 : DR \leftarrow M[AR]$

$D_2T_4 : DR \leftarrow M[AR]$

$D_3T_4 : DR \leftarrow M[AR]$

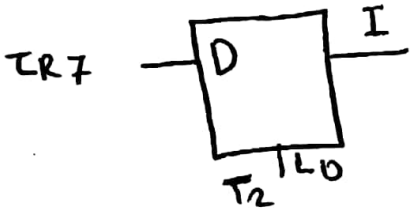


- برای نوشتن در حافظه:

$D_0T_4 : M[AR] \leftarrow AC$

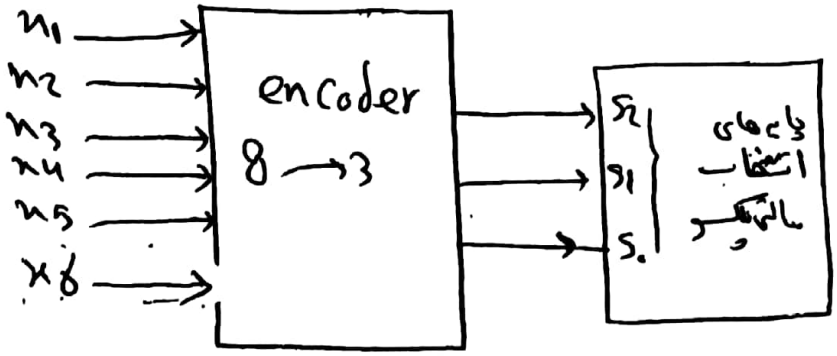
- برای فلیپ فلاپ I:

$T_2 : I \leftarrow IR(7)$



- انتخاب کننده گذرگاه:

از مدار زیر برای انتخاب گذرگاه استفاده می شود:



x_1	x_2	x_3	x_4	x_5 x_6	S_2	S_1	S_0	بایت انتخاب شده برای گذرگاه
0	0	0	0	0 0	0	0	0	هیچ کدام
1	0	0	0	0 0	0	0	1	AR ← بایت
0	1	0	0	0 0	0	1	0	PC ~
0	0	1	0	0 0	0	1	1	DR ~
0	0	0	1	0 0	1	0	0	AC ~
0	0	0	0	1 0	1	0	1	I_R ~
0	0	0	0	0 1	1	1	0	حافظه

حال باید ببینیم هر بایت چه زمانی خروجی های خود را بروی باس مشترک قرار می دهد:

- بایت AR:

- بایت PC: $T_0 : AR \leftarrow PC$

$x_2 = T_0$

- بایت DR: $D_1 T_5 : AC \leftarrow DR$

$x_3 = D_1 T_5$

- بایت AC: $D_0 T_4 : M[AR] \leftarrow AC$

$x_4 = D_0 T_4$

- بایت IR: $T_2 : AR(4-0) \leftarrow IR(4-0)$

$x_5 = T_0$

حالة:

$T_1 : IR \leftarrow M[AR]$

$IT_3 : AR \leftarrow M[AR]$

$D_2 T_4 : DR \leftarrow M[AR]$

$D_1 T_4 : DR \leftarrow M[AR]$

$D_3 T_4 : DR \leftarrow M[AR]$

$$x6 = T_1 + IT_3 + D_2 T_4 + D_1 T_4 + D_3 T_4$$