به نام خدا

نمونه سوال کامپیوتر پایه سال 99-00

سوال: ممکنه که سوالای استاد به ترتیب توی فیلم نباشه. در این صورت چی کار می‌کنیم؟
به ترتیب توی فیلم سوالا رو جواب می‌دیم. چون غیر از اون تقریبا نمی‌شه. مثلا اگر سوال اول طراحی data path رو از ما خواسته باشه، ما تا قبل از اینکه قالب دستورالعمل رو ننوشته باشیم که نمی‌دونیم چند تا ثبات IR لازم داریم. اگر توی ذهنمون حساب کنیم ممکنه اشتباه پیش بیاه و بر فرض که این اتفاق هم نیوفته به هر حال یه بار قالب رو توی ذهنمون کشیدیم. در نتیجه وقتمون سر طراحی قالب دو بار میره. در نتیجه اول برین بخش مربوط به طراحی قالب دستورالعمل رو بنویسید بعد برگردین روی بقیه‌ی بخش‌ها و ترجیحا طبق همون ترتیبی که گفتیم برین تا کارتون راحت بشه و وقتی ازتون تلف نشه.

ترتیب حل کامپیوتر پایه:

1. مشخص کردن حداقل تعداد بیت‌های مورد نیاز برای نمایش آدرس حافظه، آدرس ثبات عام منظوره، پهنای کلمه و...
2. طراحی قالب آدرس
3. طراحی مسیر داده
4. نوشتن ریز عملیات ها (و طراحی فلوچارت)
5. طراحی واحد کنترل
   a. پایه load و increment ثبات ها
   b. پایه read و write حافظه اصلی
   c. پایه های کنترلی ALU
   d. پایه های کنترلی گذرگاه

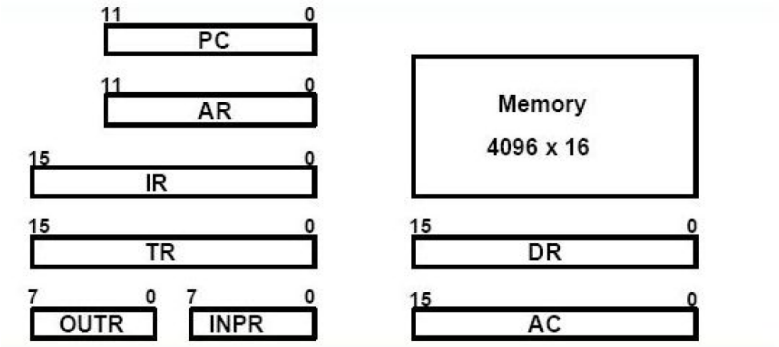
مثال)
حافظه اصلی با ابعاد 1**k Byte** و ثبات های عام منظوره ۸ بیتی **A**، **B**، **C** و **D** است. عدد بلافصل ۴ بیتی می‌تواند باشد. به موارد زیر پاسخ دهید:
سوال: اگر بخوایم دو تا باس داشته باشیم، پهنای باس آدرس و داده چند بیته؟
آدرس:10 بیت
داده:8 بیت
سوال: حداقل چند بیت لازم داریم برای مشخص کردن ثبات‌های عام منظوره؟
آدرس ثبات: 2 بیت
دقیقشو بخوایم بگیم قالب آدرس رو می‌کشیم تا بفهمیم چند تا بیت فضا برای آدرس‌دهی عام منظوره‌ها داریم. اگر ۴ تا فضا رو داشتیم که بهتر چون دیگه دیکدر هم لازم نداره اما اگه نداشتیم باید با ۲ بیت کار رو انجام بدیم.

دستورات پردازنده	توضیحات
جمع **1Op** و **2Op** و ذخیره در **1Op** **1Op** داده از حافظه و **2Op** بلافصل 4 بیتی	**< ADD <op1>, <op2**
ذخیره سازی در حافظه **1op** آدرس حافظه و **2Op** بلافصل 4 بیتی	**STR <op1>, <op2>**
پوش کردن به پشته **Op**: بلافصل 4 بیتی یا آدرس ثبات عام منظوره	**Push <op>**

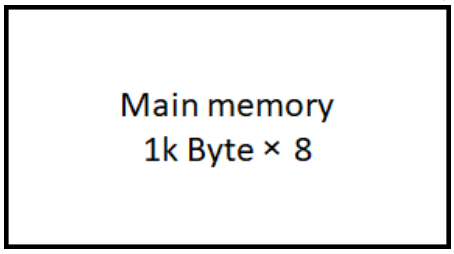
سوال: از روی دستورهای بالا بگید که کلا چند تا دستور داریم؟
کلا 4 تا دستور میشه

أ. ثبات های کنترلی و داده‌های لازم در طراحی این پردازنده (۱ نمره)
سوال: منظور این سوال چیه؟
منظور سوال اینه قبل از متصل کردن مدار و سیم کشی موادی که در مسیر داده می‌خوایم استفاده کنیم رو مشخص کنیم. چه ثبات‌هایی داریم و هر کدوم چند بیت هستن.
مثل همچین چیزی تو جزوه‌ی استاد:

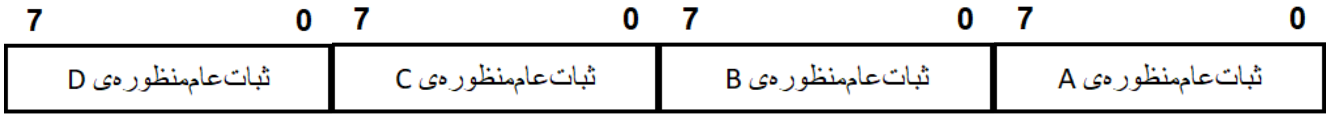
با توجه به Instruction Set درمورد حافظه و ثبات‌های مورد نیازمان تصمیم گیری می‌کنیم :



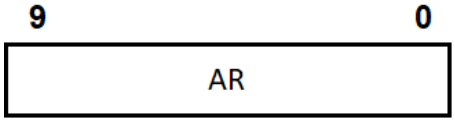
سوال: خب حالا جوابش برای این سوال چی میشه؟
گفتیم که اول اگر بریم سراغ طراحی قالب دستورالعمل کارمون راحت‌تره. چون تا اونو طراحی نکرده باشیم نمی‌دونیم چند تا ثبات IR نیاز داریم.
(بعد از برگشت از مرحله‌ی ث میایم اینجا رو کامل کنیم)



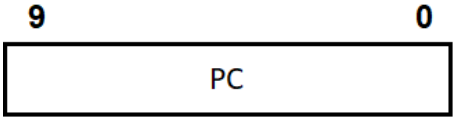
این رو که اطلاعاتش رو از روی سوال داشتیم.



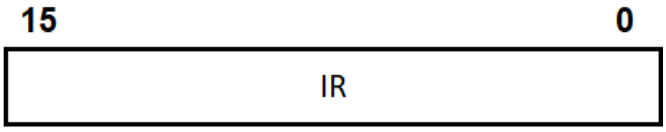
در اطلاعات مسئله گفته شد چهار تا ثبات عام منظوره داریم که ۸ بیتی هستن. از روی همین متوجه میشیم که ثبات های عام منظوره ی دیتا هستند.



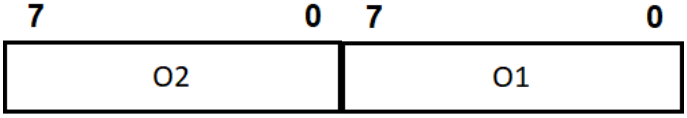
ثبات AR که مخفف address reg بود توی خودش قراره یه آدرس نگهداری کنه. ما هم که حافظه‌مون یک کیلوبایت = دو به توان ۱۰ ردیف داره. در نتیجه ثبات‌هایی که قراره آدرس نگهداری کنن، ۱۰ بیتی میشن.



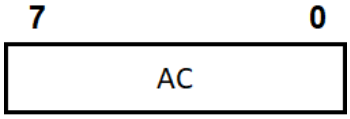
خب اینم که program counterمونه و توی رم بالا و پایین میره و آدرس خطهای برنامه رو تو خودش ذخیره میکنه. یعنی ماهیت اطلاعاتش آدرسه و در نتیجه ۱۰ بیتیه.



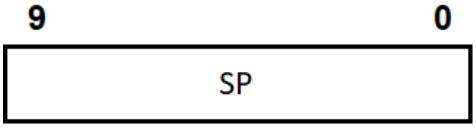
خب از اونجایی که تو قالب دستورالعمل رسیدیم به اینکه به دو تا کلمه نیاز داریم، در نتیجه به ۱۶ بیت برای نگهداری هر دستور نیاز داریم. حالا این کار رو می‌تونیم به کمک دو تا ثبات 1IR و 2IR که جفتشون ۸ بیتی هستن انجام بدیم؛ هم می‌تونیم یک ثبات ۱۶ بیتی برداریم. دقت* اندازه IR ضریب صحیحی از پهنار کلممونه.



این دو تا ثبات رو برای نگهداری ورودی‌های ALU میذاریم. ALU هم که قراره دو تا دیتا با هم جمع کنه در نتیجه جفتشون ۸ بیتی میشن.

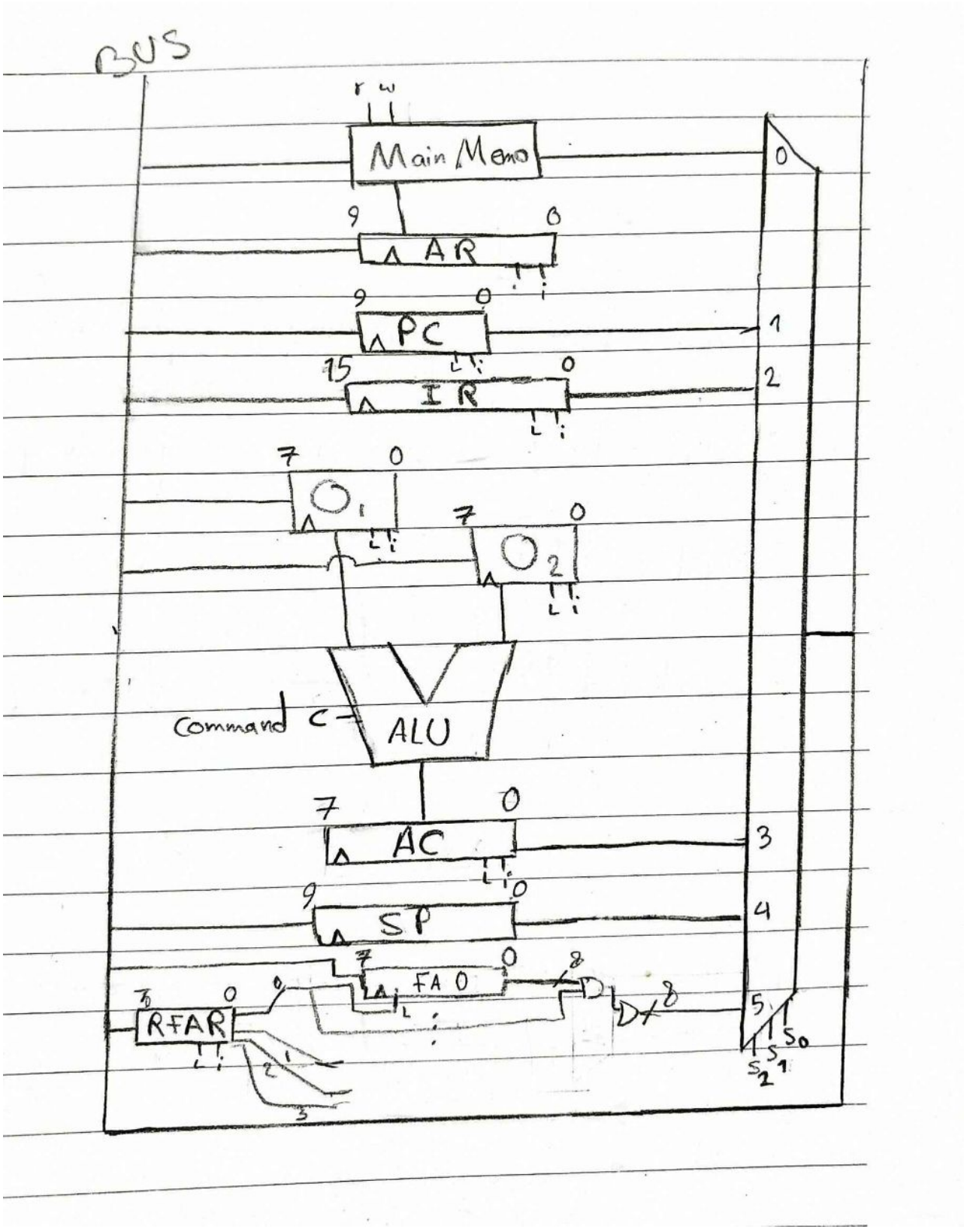


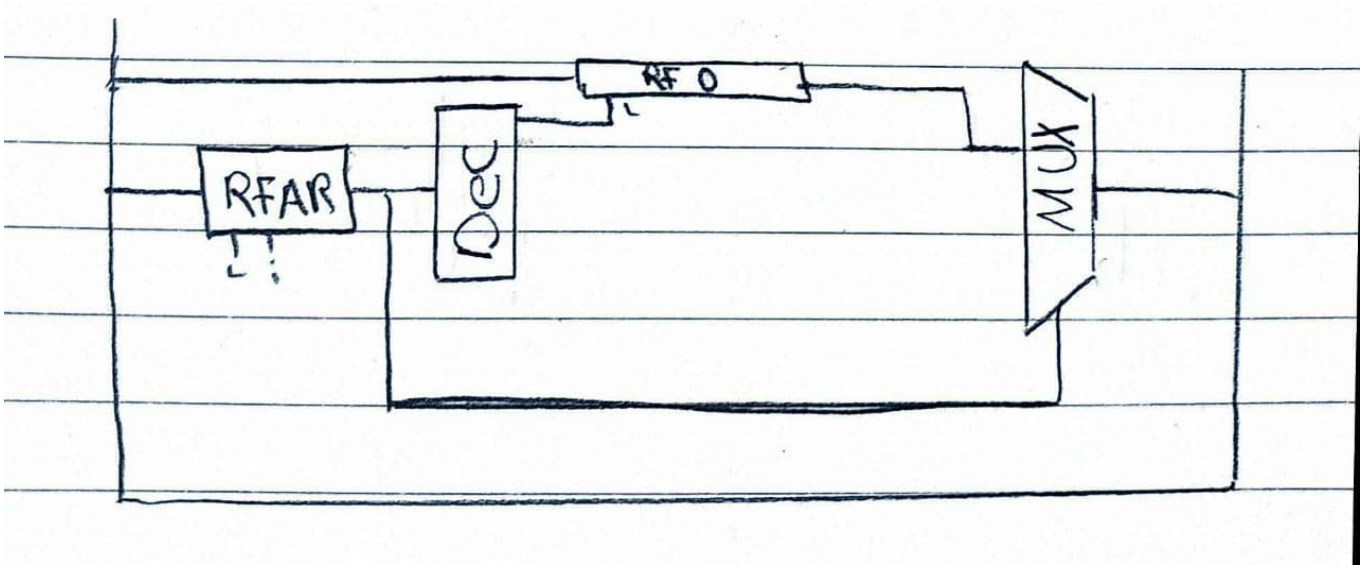
این ثبات می‌تونه خروجی ALU رو برامون نگهداره و چون یک دیتا تو خودش نگهداری میکنه ۸ بیتیه.



مخفف stack pointer. پوینتر یعنی به یه آدرسی اشاره می‌کنه برای همین ۱۰ بیتیه.

ب. طراحی مسیر داده پردازنده (۲ نمره)


BUS
r w
Main Memo
9
0
AR
9
0
PC
15
0
IR
7
0
O1
7
0
O2
Command C
ALU
7
0
AC
9
0
SP
7
0
FA 0
8
RFAR
3
0
0
1
2
3
8
0
1
2
3
S0
S1
S2



در حالت کلی ثبات های عام منظوره به صورت بالا هستند.

ت. ریزعملیات مربوط به هر دستور را بنویسید (۲ نمره)

Instruction fetch:

T0: AR← PC

T1: IR[0:8] ← M[AR], PC+=1

T2: AR← PC

T3: IR[8:16] ← M[AR], PC+=1

(بازه‌ها شبیه زبان پایتون نوشته شده. مثلا IR[0:8] یعنی بیت ۰ تا ۷ ثبات IR)

Instruction decode:

T4: decode IR[14:16], SP ← (1111111111)

(در حالت کلی sp را داخل مراحل الگوریتم فون نیومن نمی نویسیم که هر بار ریست شود. آن را قبل از شروع الگوریتم مقدار دهی می کنیم.)

ADD:

T5.D0: AR← IR[4:14]

T6.D0: O1← M[AR]

T7.D0: RFAR← IR[0:4]

T8.D0: O2← RF[RFAR]

T9.D0: AC← O1+O2

T10.D0: M[AR] ← AC, sc ← 0

STR:

T5.D1: AR ← IR[4:14]

T6.D1: M[AR] ← IR[0:4], sc ← 0

PUSH immediate:

T5.D2: AR ← SP

)به طور استاندارد فقط AR به حافظه متصل است و نمیتوان sp را هم به آن متصل کرد و حتما باید ابتدا sp را داخل AR ریخت(

T6.D2: M[AR] ← 0000:IR[0:4] , SP -= 1, sc ← 0

PUSH reg:

T5.D3: RFAR ← IR[0:4]

T6.D3: AR ← SP

T7.D3: M[AR] ← RF[RFAR], SP -= 1, sc ← 0

ث. طراحی قالب دستورالعمل (طول دستورالعمل و مشخص سازی فیلدهای مختلف آن) (۲ نمره)

چهار دستور العمل داریم ← 2 بیت برای opcode

2 بیت برای آدرس دهی ثبات‌های عام‌منظوره

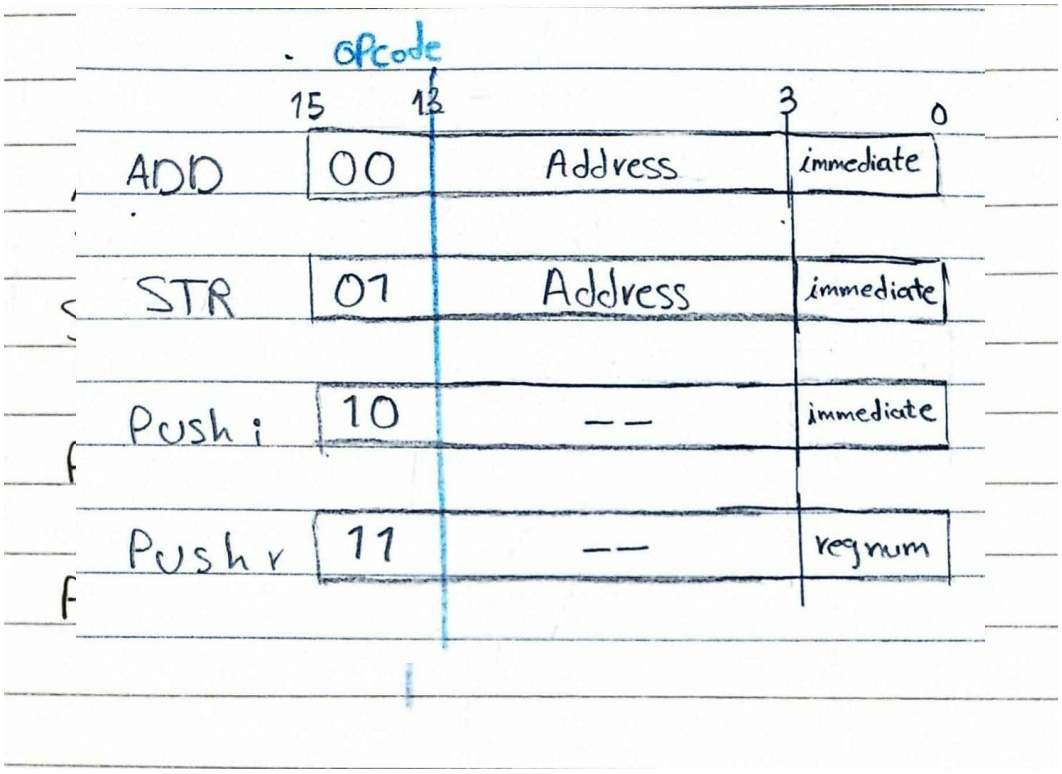
10 بیت آدرس حافظه

⇐ طولانی ترین دستور ADD است

حداقل تعداد بیت مورد نیاز؟ آپکد + داده از حافظه + 4 بیت بلافصل ⇐ حد اقل 16 بیت

میدانیم طول دستور مضربی از پهنای کلمه است پس دستور 16=8*2 بیتی در نظر می‌گیریم.

توضیحات	دستورات پردازنده
ADD <op1>, <op2 >	جمع Op1 و Op2 و ذخیره در Op1 Op1 داده از حافظه و Op2 بلافصل 4 بیتی
STR <op1>, <op2>	ذخیره سازی در حافظه op1 آدرس حافظه و Op2 شماره ثبات
Push <op>	پوش کردن به پشته Op: بلافصل 4 بیتی یا آدرس ثبات عام منظوره

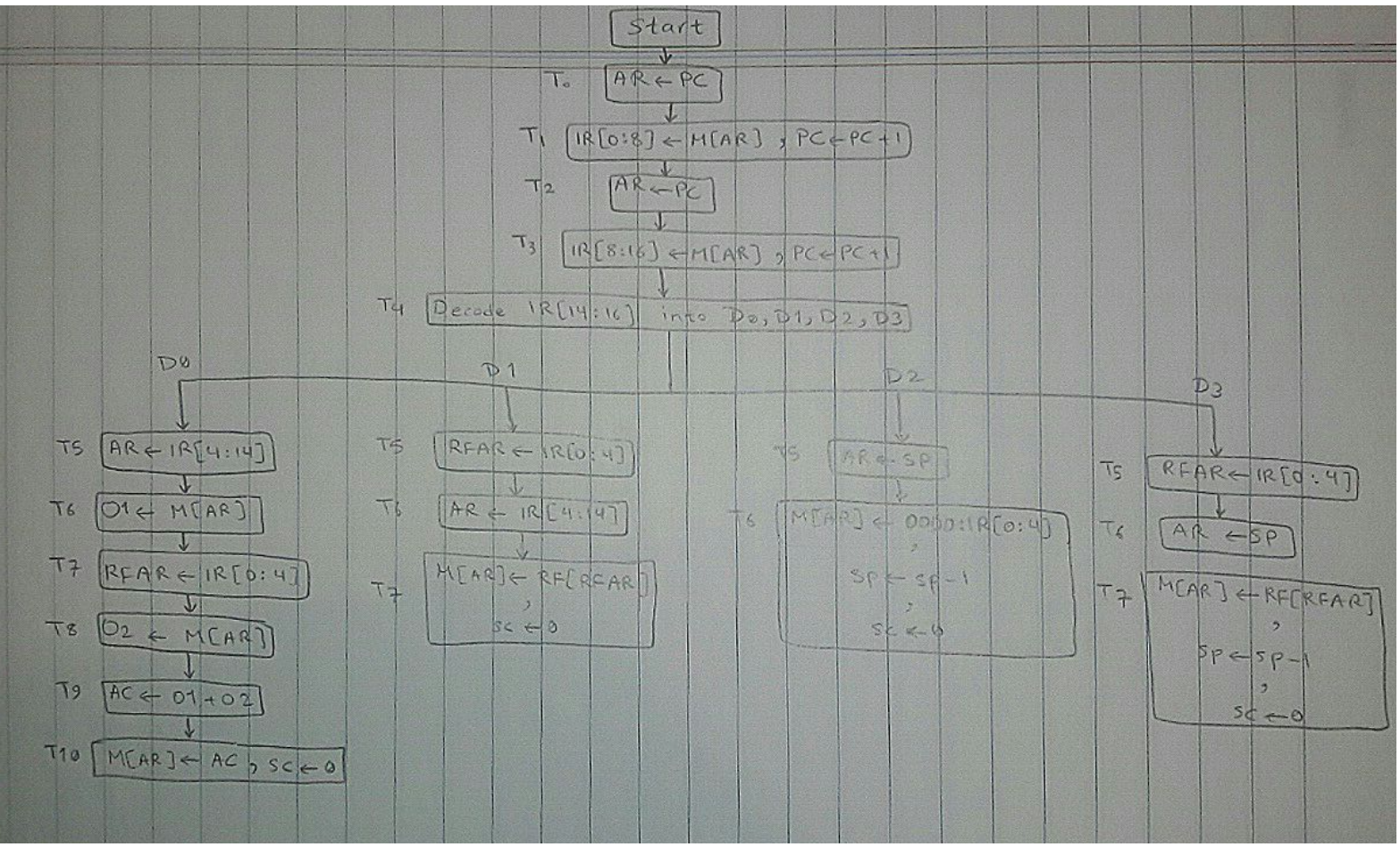


ج. روندنما (فلوچارت) اجرای دستورالعمل را در این پردازنده ترسیم کنید (۲ نمره)
ح. طراحی واحد کنترل پردازنده (مدارات ورودی پایه **Multiplexer Load**ها و بقیه مدار های کنترلی لازم) (۲ نمره)

★ پایه های کنترلی ثبات ها

- Increment و decrement : ترجیحا این پایه‌ها رو برای همه‌ی ثبات‌ها تعریف کنین حتی اگه تو ریزعملیات‌ها برای ثباتی نیازش نداریم. اما حتما برای ثبات PC پایه‌ی Increment رو نیاز داریم. و پایه‌ی decrement رو هم برای ثبات SP باید قطعا داشته باشیم.

increment_PC=T1+T3



decrement_SP=T6D2+T7D3
clear_sc=T7D3+T6D2+T6D1+T7D0

- Load :

- پایه‌ی لود برای ثبات‌های عام‌منظوره: این پایه‌ها تو datapath طراحی شدن. چهار بیتی که برای مشخص کردن یه ثبات عام منظوره تو قالب دستورالعمل مشخص شده، هر بیتش یه پایه‌ی لود برای هر کدوم از این ثبات‌هاست.
- ثبات AR: پایه‌ی لود می‌شه تمام زمان‌هایی که تو ریزعملیات‌ها، ثبات AR در حال مقدارگرفتنه.

AR_Load: T0 + T2 + T5.D0 + T5.D1 + T5.D2 + T6.D3

- ثبات PC:

PC_Load: 0

- ثبات IR:

IR_Load: T1 + T3

- ثبات 1O:

O1_Load: T6.D0

- ثبات 2O:

O2_Load: T8.D0

- ثبات AC:

AC_Load: T9.D0

- ثبات SP:

SP_Load: T4

★ پایه های کنترلی Main Memory

- Read:

M_Read: T1 + T3 + T6.D0

- Write:

M_Write: T10.D0 + T6.D1 + T6.D2 + T7.D3

★ پایه های کنترلی ALU

- پایه‌ی COMMAND: از اونجا که فقط یک عملیات داریم که به ALU نیاز داشته باشه، این پایه تک بیتی میشه و باید هر زمان که قراره جمعی اتفاق بیوفته، فعال بشه.

COMMAND: T9.D0

★ پایه کنترلی BUS

- دیدیم که ۶ تا ورودی مختلف به گذرگاه کلی داریم پس به ۳ بیت برای انتخاب هر کدوم از اون حالت‌ها نیاز داریم. بیت‌های 0S تا 2S.
برای این به یک دیکدر نیاز داریم که این سه بیت رو تولید کنه تا در مواقع لازم به درستی و به موقع فعال بشن. برای تولید ۳ بیت به یک دیکدر 8:3 احتیاج داریم. ورودی‌های این دیکدر رو با 0x تا 7x نشون میدیم.

	x7 x6 x5 x4 x3 x2 x1 x0	S2 S1 S0
Memory	00 000001	000
PC	00 000010	001
IR	00 000100	010
AC	00 001000	011
SP	00 010000	100
MUX	00 100000	101

x0: T1 + T3 + T6.D0
x1: T0 + T2
x2: T5.D0 + T7.D0 + T5.D1 + T6.D1 + T6.D2 + T5.D3
x3: T10.D0
x4: T5.D2 + T6.D3
x5: T8.D0 + T7.D3
x6: 0
x7: 0

خ. برنامه ای به زبان اسمبلی بنویسید که 1 تا 3 را باهم جمع کند و در حافظه با آدرس 64 ذخیره کند. (۱ نمره)

```
01 0001000000 0000 str [64], 0
00 0001000000 0001 add [64], 1
00 0001000000 0010 add [64], 2
00 0001000000 0011 add [64], 3
```