

Computer Architecture

HW4 Report

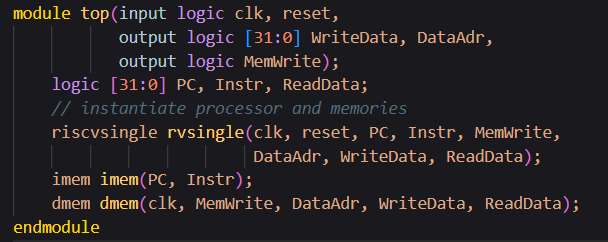
G14 (Mohammad Bahrami & Navid Raeiszadeh)

**سوال دوم:**

برای اضافه کردن سه دستور addi , jal, jalr کافی است datapath وcontroller رو مانند شکل سوال یک تغییر داد.

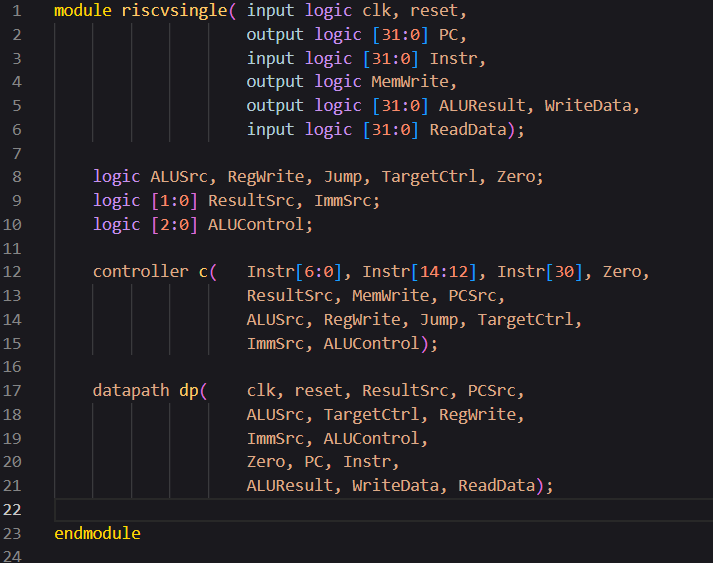
کد هایی که در منبع هریس برای مدل HDL پردازنده single cycle بود رو کپی کرده و تغییرات لازم رو اعمال میکنیم.

**ماژول top :**



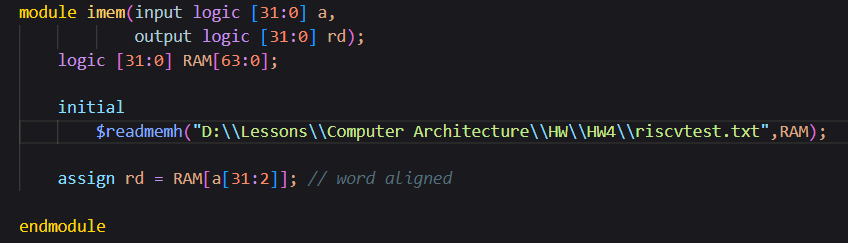
بدون تغییر باقی میماند.

**ماژول riscvsingle :**



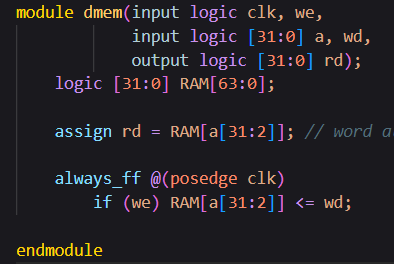
سیگنال TargetCtrl که در سوال یک به عنوان سلکت مالتیپلکسر اضافه شده بود را به کد نیز اضافه میکنیم.

**ماژول imem :**



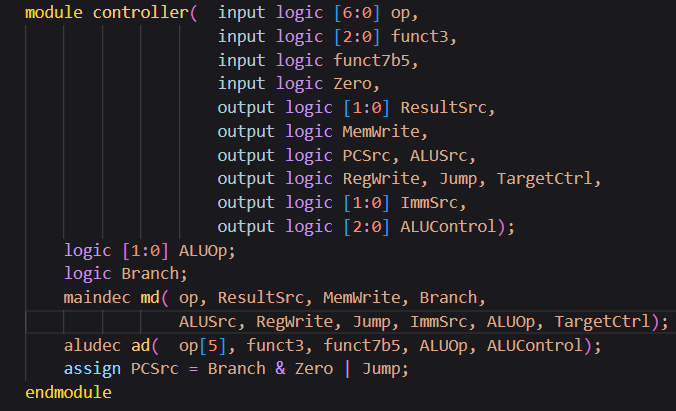
آدرس فایل تکست ماشین کد دستورالعمل هارا در readmemh قرار میدهیم.

**ماژول dmem:**

****

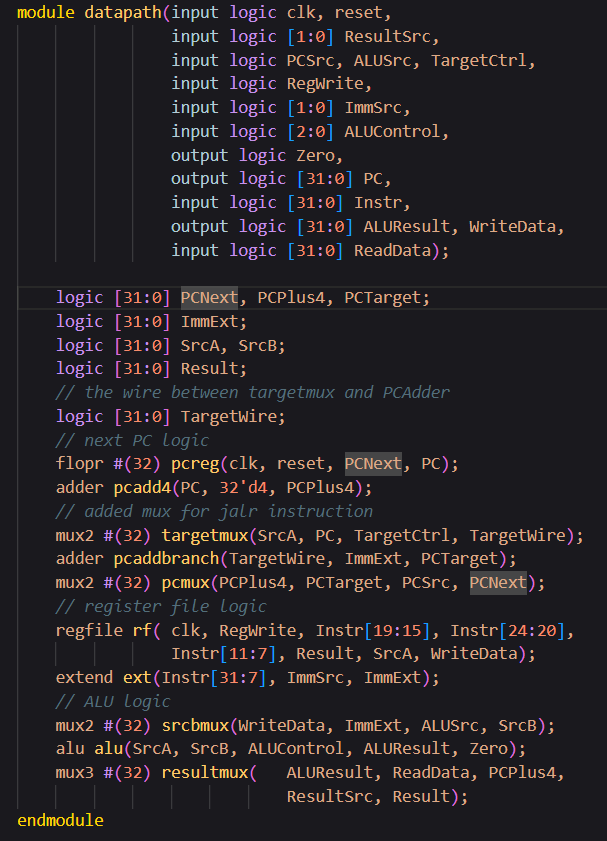
بدون تغییر.

**ماژول controller :**



سیگنال TargetCtrl را به main decoder میدهیم.

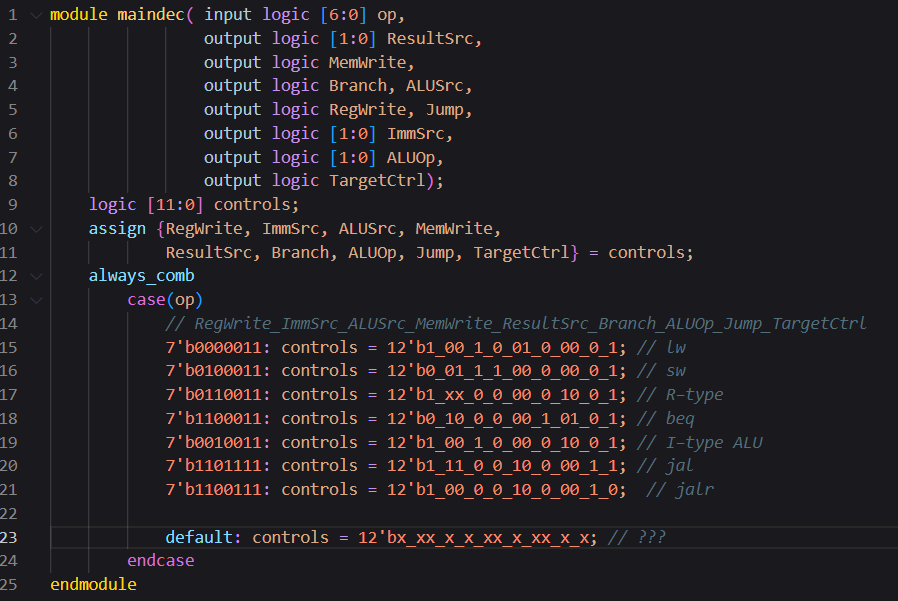
**ماژول datapath :**

****

باس TargetWire بعد از مالتیپلکسر قرار دارد و وارد adder pc target میشود

تغییر اصلی در دیتاپث اینجا اعمال شده که بین PC و SrcA یک مقدار برای جمع شدن با آفست انتخاب میشود که SrcA برای دستورjalr هست.

**ماژول maindecoder** :



سیگنال TargetCtrl را به controls کانکت میکنیم و که در این صورت سیگنال controls دوازده بیتی خواهد شد و بیت آخر مربوط به TargetCtrl خواهد بود.

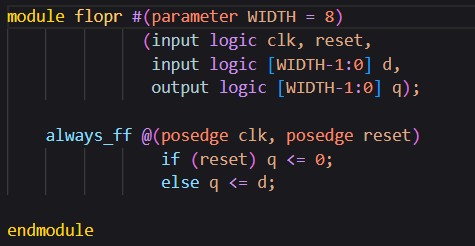
جدول درستی در سوال یک آورده شده است.

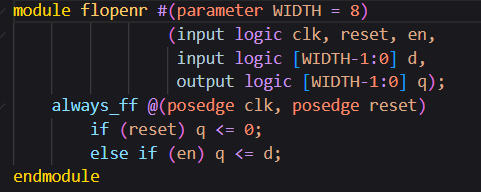
**ماژول aludec :**

****

بدون تغییر.

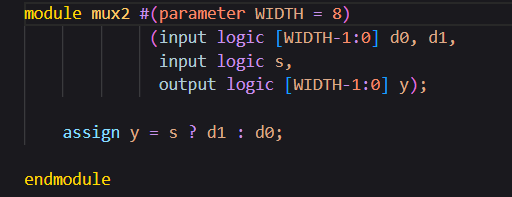
**ماژول flopr** و **flopenr**:

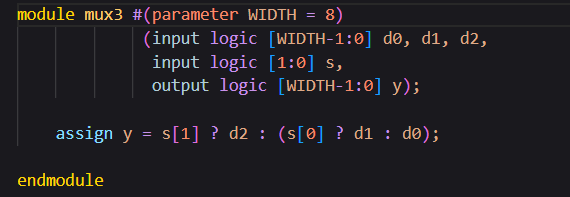




هر دو بدون تغییر.

**ماژول های mux2, mux3** :

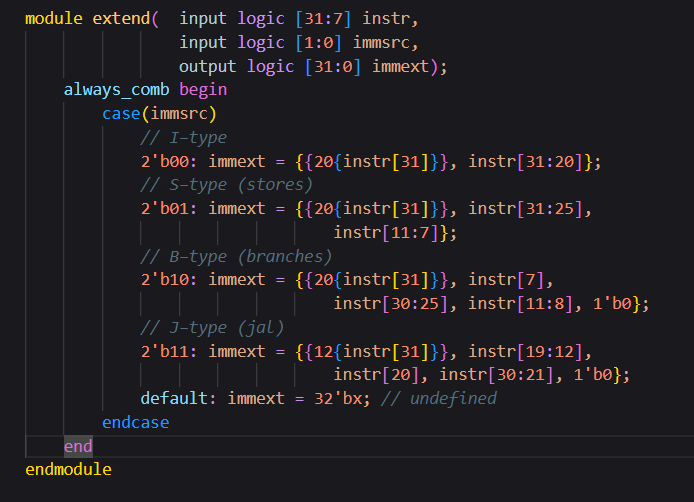




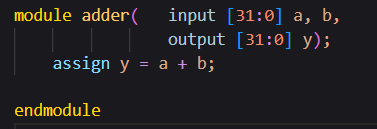
بدون تغییر.

**ماژول extend** :

بدون تغییر

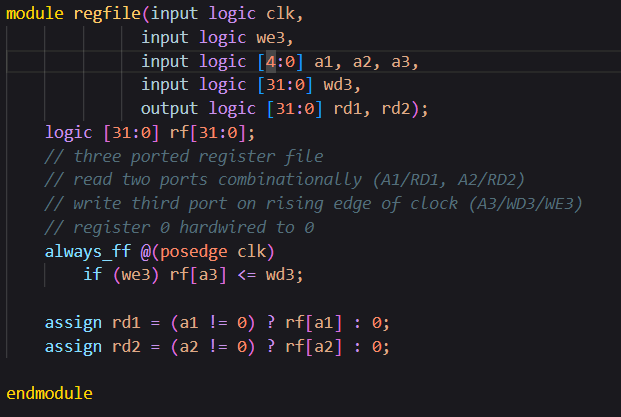


**ماژول adder :**



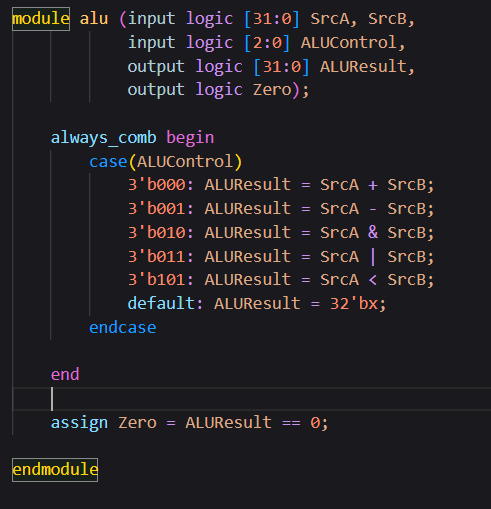
بدون تغییر.

**ماژول regfile** :



ورودی های a1.a2,a3 باید 5 بیتی باشند که در کد کتاب 6 بیتی بودند.

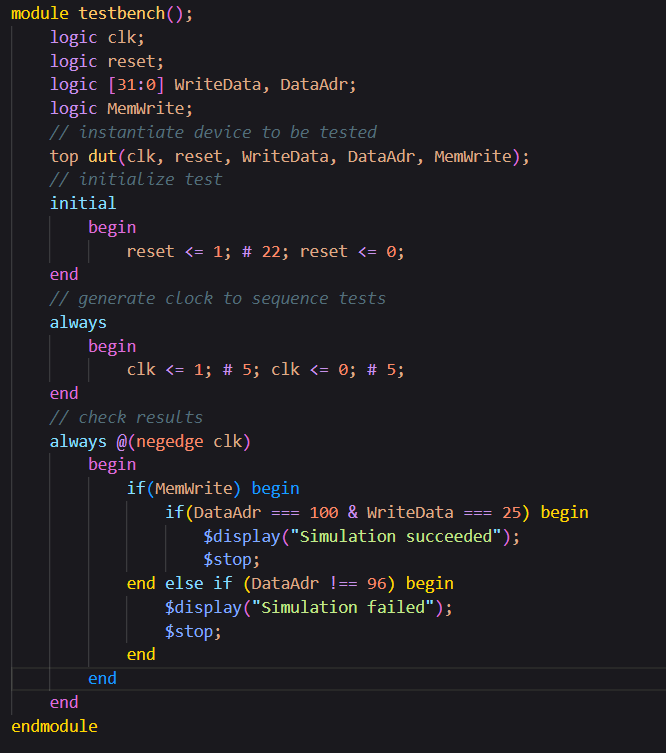
ماژول alu :



5 دستور مورد نیاز پردازنده که در جدول کتاب هم آمده است را پیاده سازی میکنیم.

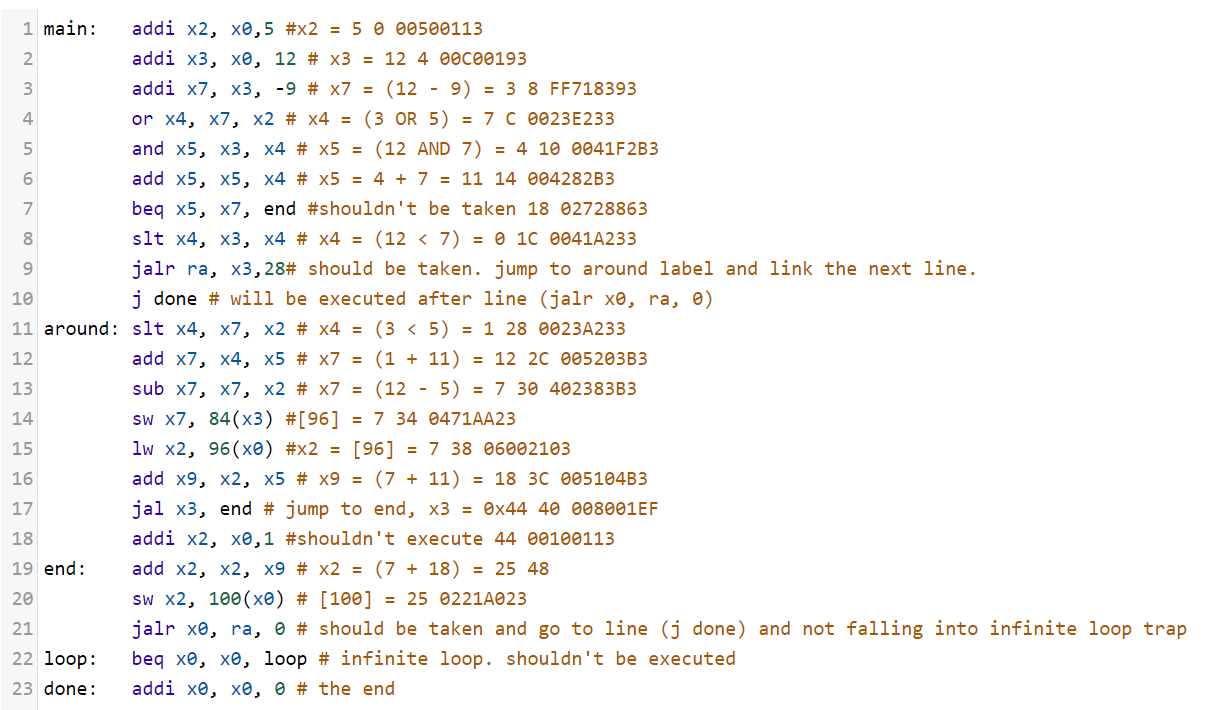
**تست بنچ:**

در این تست فایل riscvtest.txt که حاوی ماشین کد کد اسمبلی مورد نظر ما میباشد را خوانده و اگر پس از اتمام خواندن در خانه 100 مموری مقدار 25 رایت شده باشد موفقیت آمیز خواهد بود.



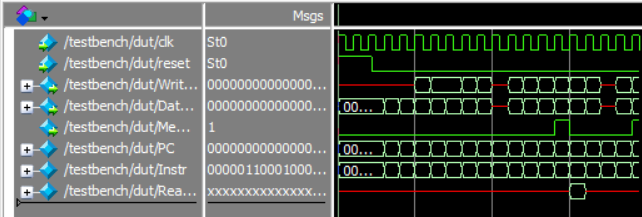
تغییری نسبت به کد کتاب نداشته است .

**کد اسمبلی:**

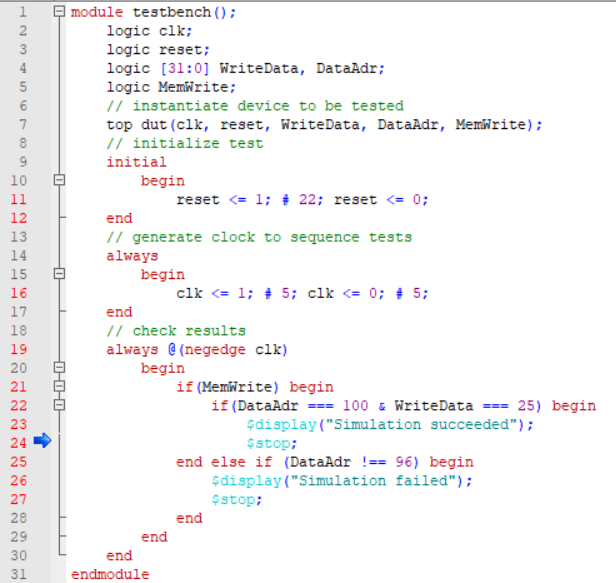
****

در اینجا در خط 9 م برنامه با استفاده از دستور jalr مقدار رجیستر x3 که 12 میباشد را با 28 جمع کرده تا به PC به مقدار 40 برسد که آدرس شروع لیبل around است و ازین به بعد تا خط 21 مشابه کد کتاب است و پس از رایت کردن مقدار 25 در خانه 100 مموری با استفاده از دستور jalr x0, ra, 0 به خط 10 پرش کرده و از آنجا به لیبل done جامپ میکنیم و اگر jalr به درستی اجرا نشود به لیبل loop میرسیم که یک حلقه بی نهایت اجرا میشود.

**شکل موج در سیمولیشن:**



**توقف اجرا برنامه و موفقیت آمیز بودن:**

****