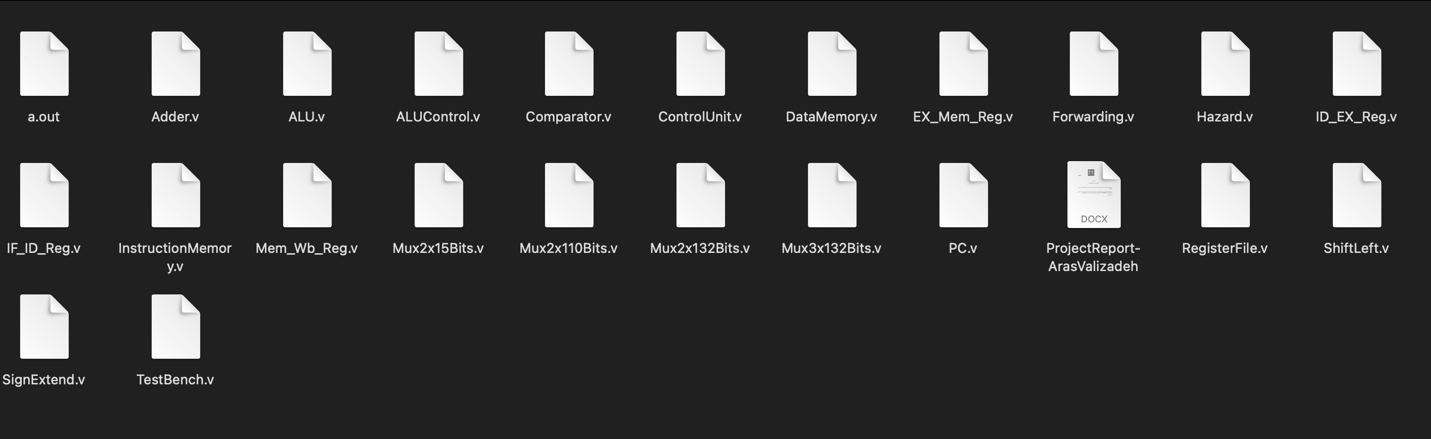
401243095  آراس ولی‌زاده

به نام خدا

گزارش پروژه معماری کامپیوتر

در این پروژه با ساختار پایپ لاین که در طول تدرس آشنا شده بودیم با استفاده از وریلاگ آن را پیاده سازی می‌کنیم .

ساختار کلی بدین صورت است که تمامی کامپوننت های تعریف شده با وریلاگ در یک فایل جمع شده و تمامی کامپوننت ها در فایل testbench گرده هم می‌آوریم و پردازنده رو در آن قسمت سر هم می‌کنیم .

ساختار کلی :  
  
 

برای بخش ابتدایی پروژه ، باید دستور های مورد نظر را در حافظه دستورات نگه‌داری کنیم برای این کار در حافظه دستورات به ازای pc+4 که به ایندکس های مختلف مموری اشاره می‌کند دستورات را بر اساس opcode های پیش فرض تعریف کرده و درون حافظه دستورات نگه‌داری می‌کنیم .

در ابتدا دو دستور load ، برای اینکه مقادیر پایه f(0) و f(1) را به رجیستر های موجود بیاوریم ، سپس تا زمانی که به پنجمین جمله تابع فیبوناچی برسیم و آن را محسابه می‌کنیم .

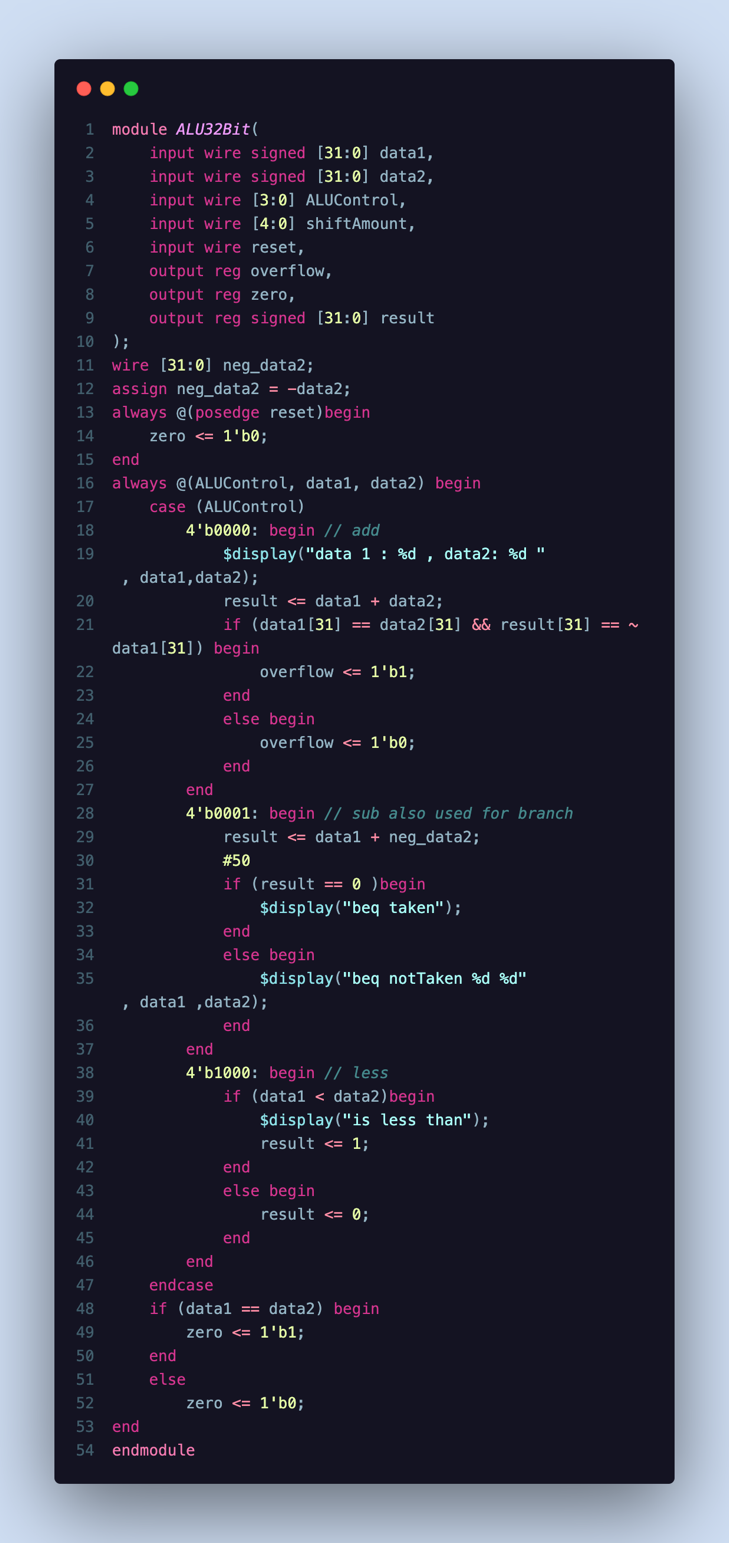


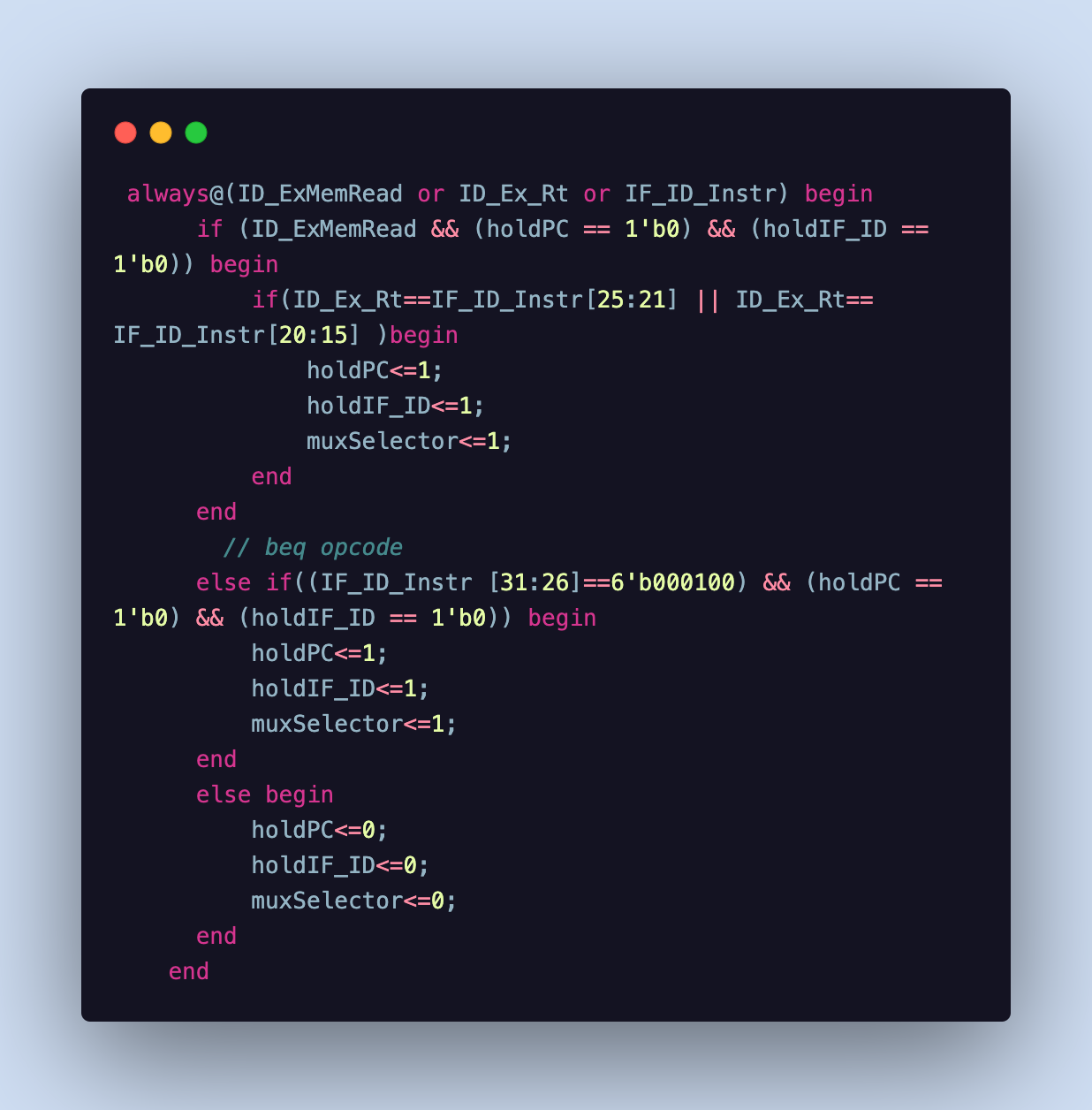
دستورات طبق اپ کد هایی که دارند به بخش کنترل یونیت فرستاده شده و در نهایت سیگنال های کنترلی متناسب با هر دستور ساخته می‌شود .



سیگنال های کنترلی ساخته شده متناسب با هر دستور در به صورت متوالی درون رجسیتر های بین استیج ها هدایت می شوند . سینگال کنترلی واحد ای ال یو به واحد کنترلی آن پاس میدهیم و برای ساخت سیگنال کنترلی خود ای ال یو متناسب با نوع دستور و ۶ بیت انتهایی دستور که فانکشن می‌باشد ، سینگال کنترلی را برای ای ال یو میسازیم .



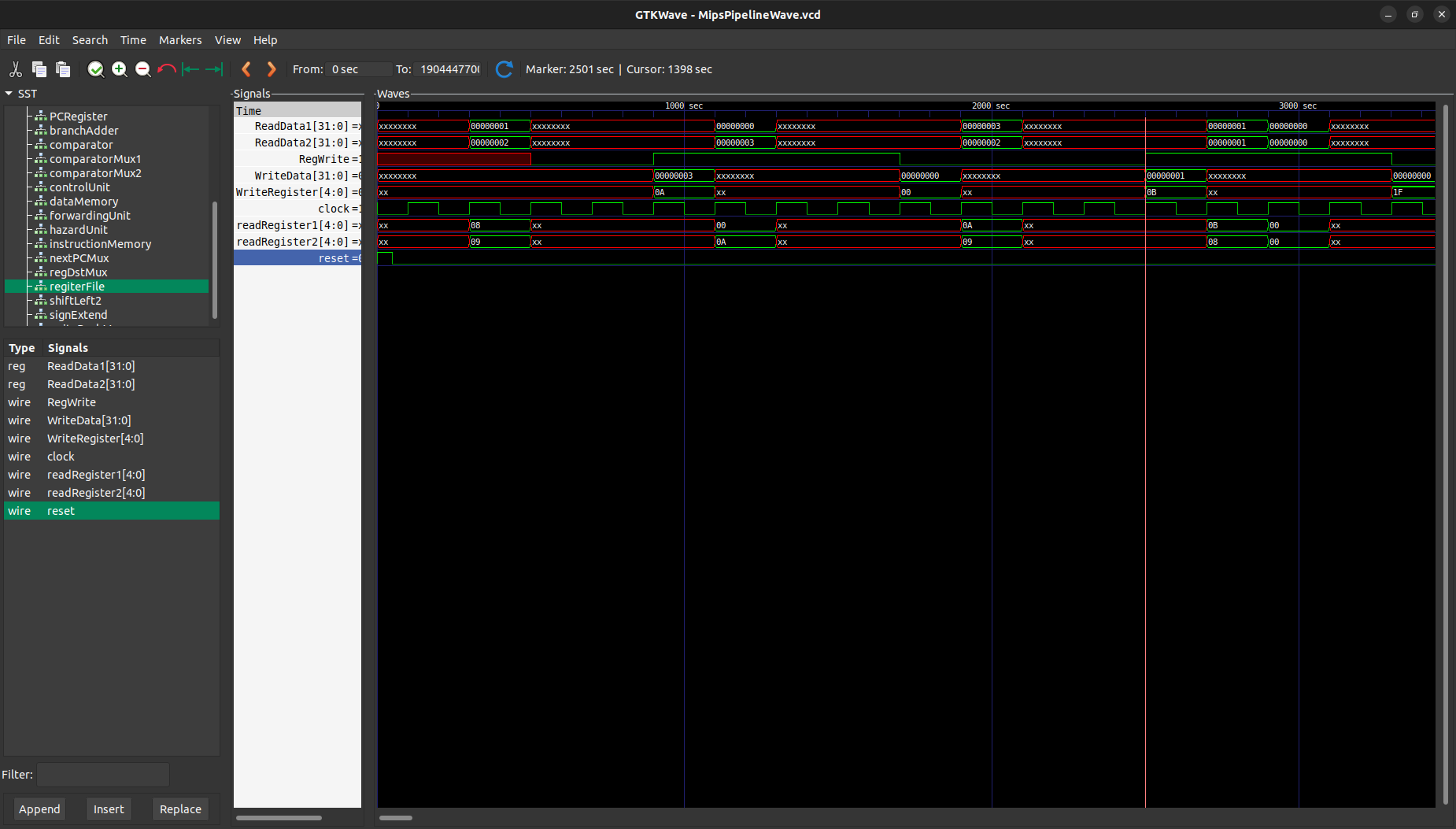
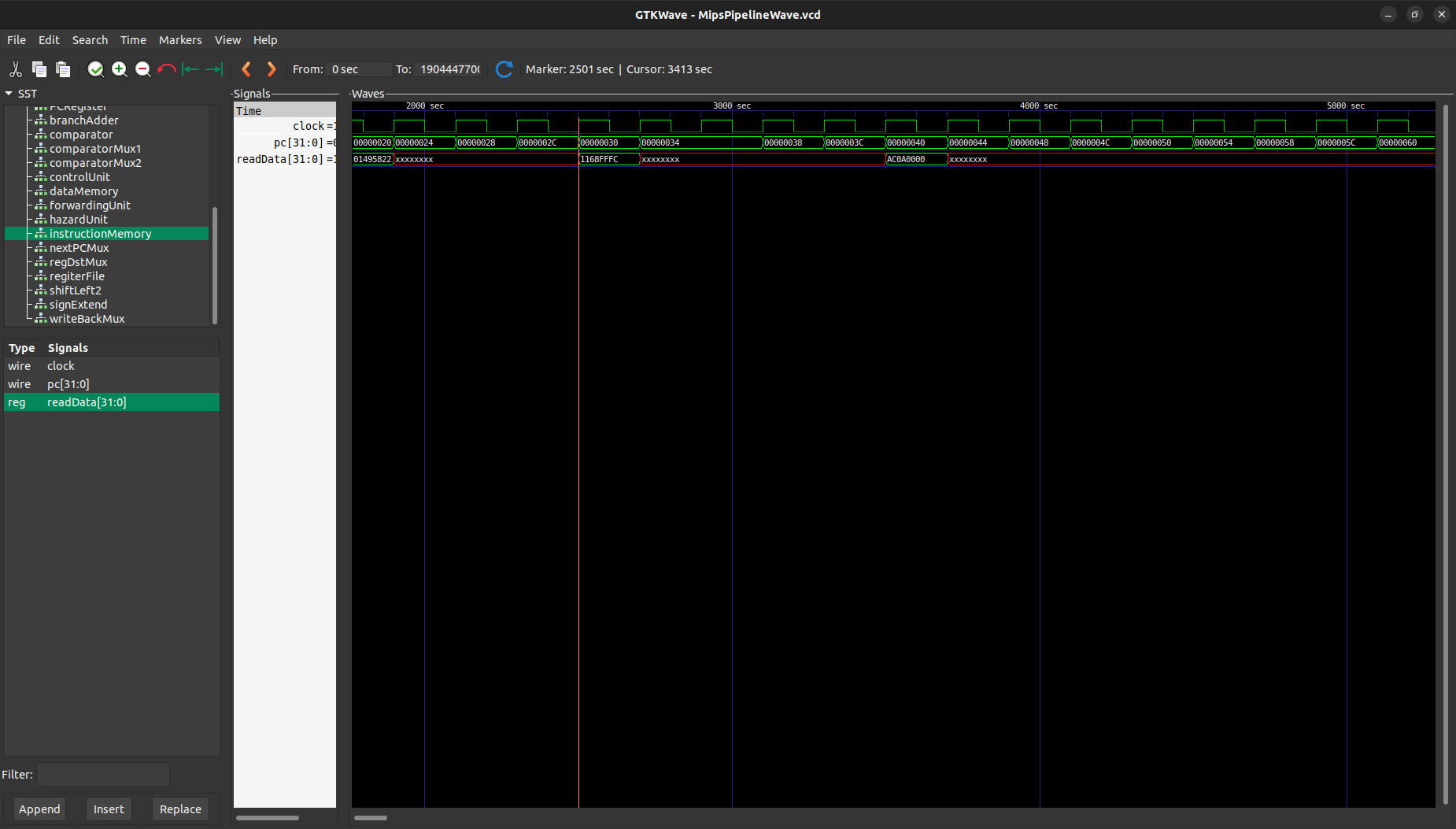
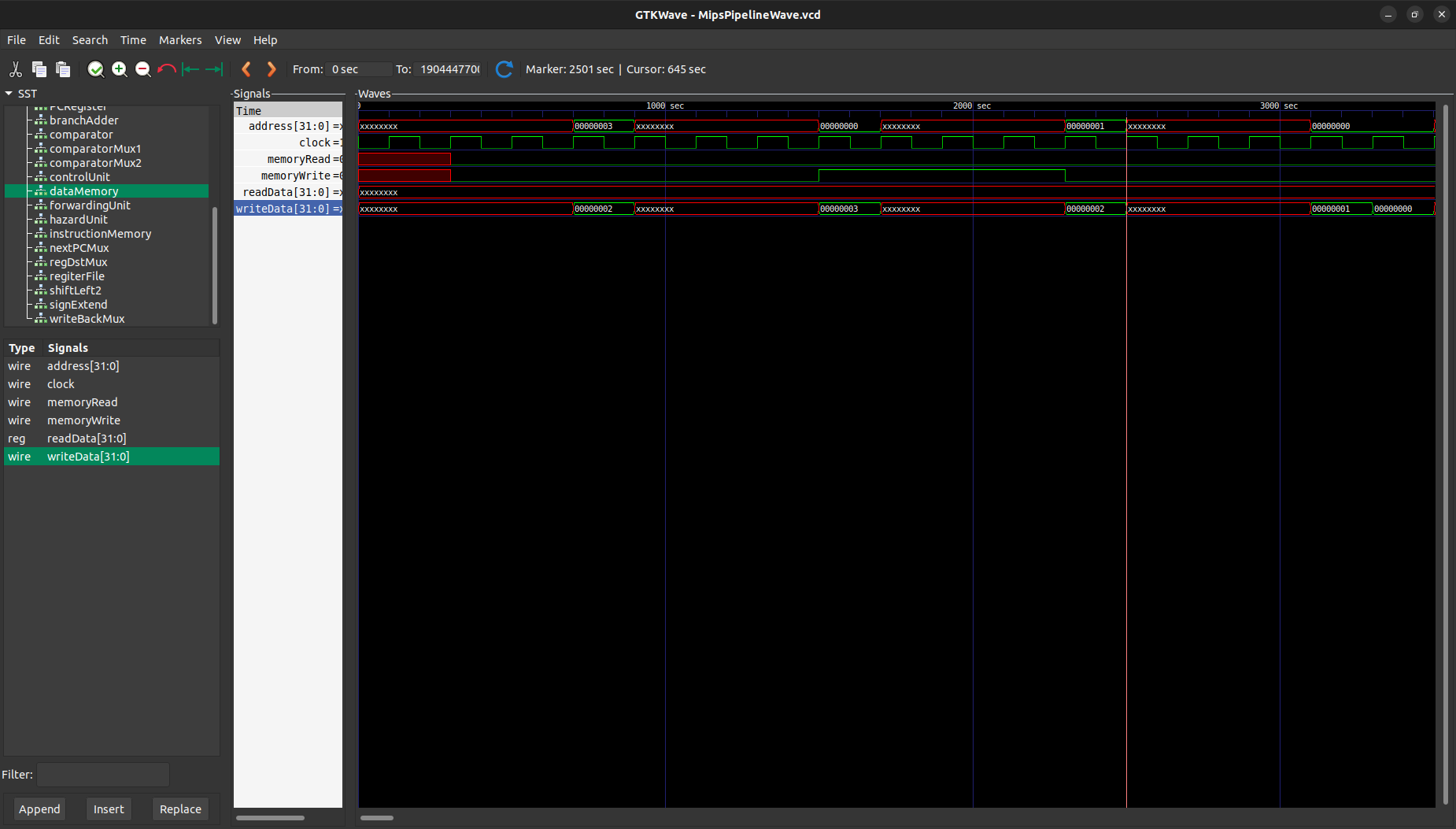
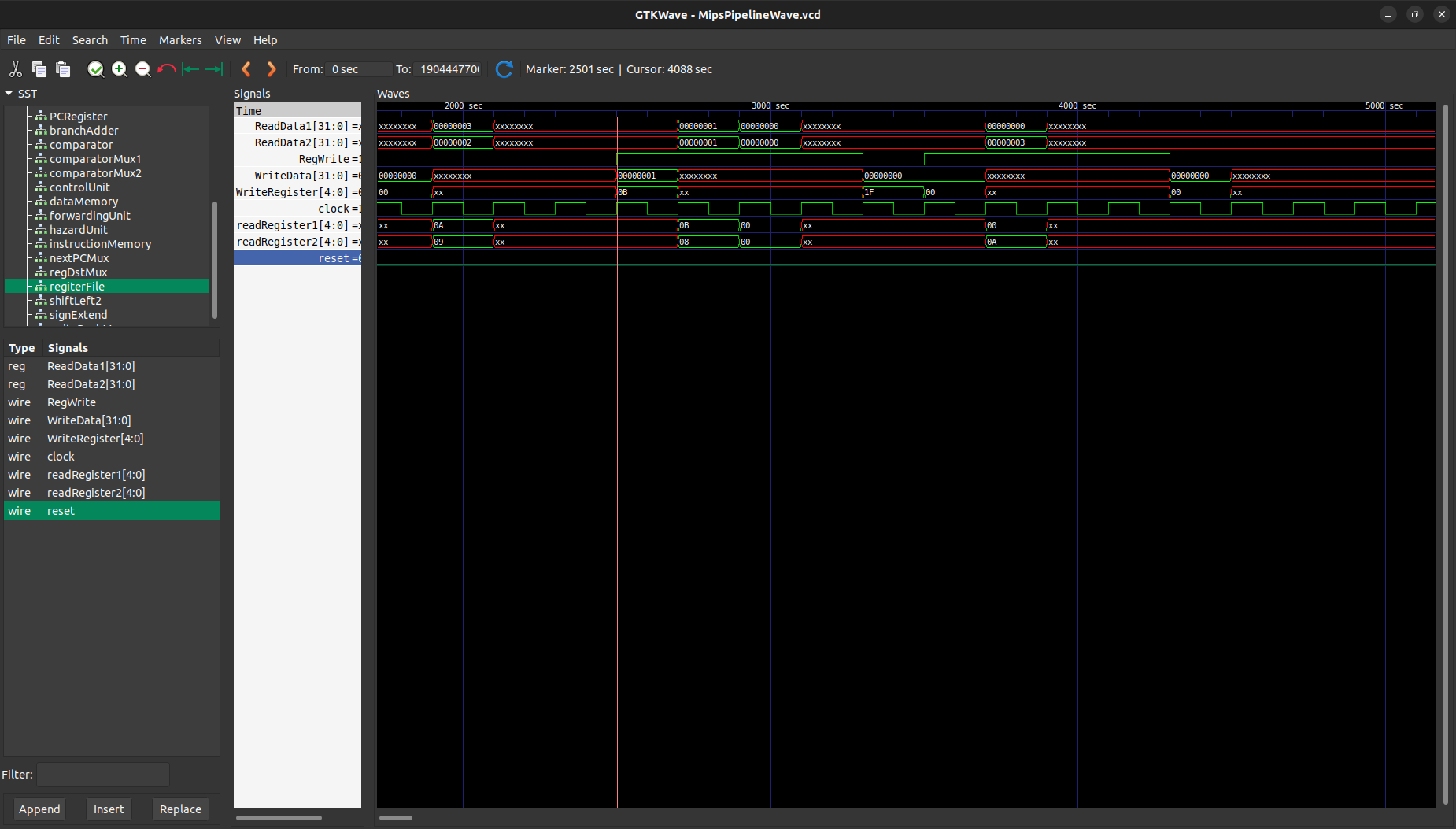
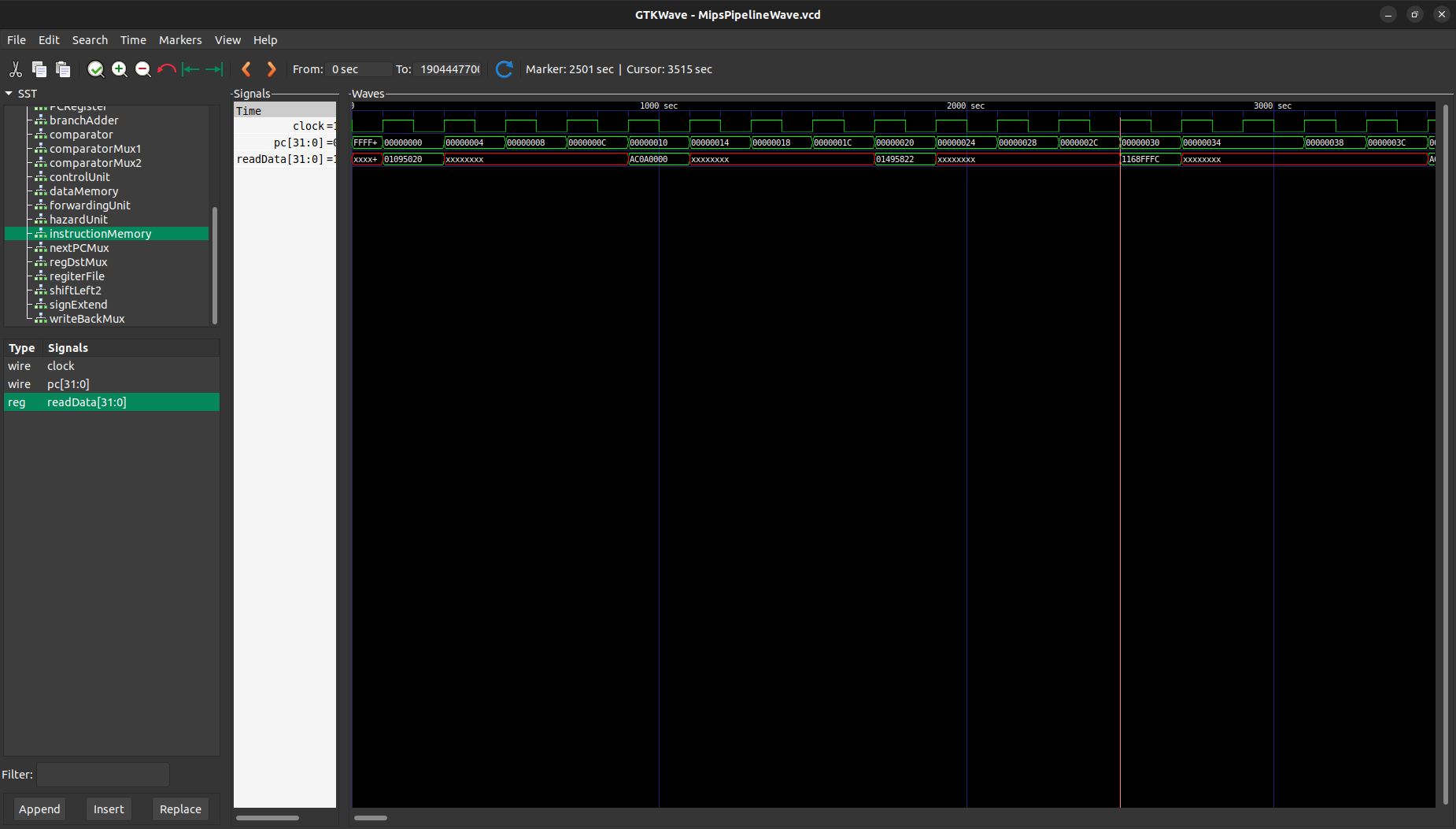
در واحد کنترلی ای ال یو با توجه به سیگنال ورودی عملی که ال ای یو باید انجام دهد را مشخص میکنیم دو دستور برنچ و کم کردن دو عدد از یکدیگر با هم همپوشانی دارند در نتیجه با یک سینگال یکسان که در ۴ بیت ساخته می ‌شود این دو دستور را هندل میکنیم ، ای ال یو به شکل زیر پیاده سازی شده است :  
  
  


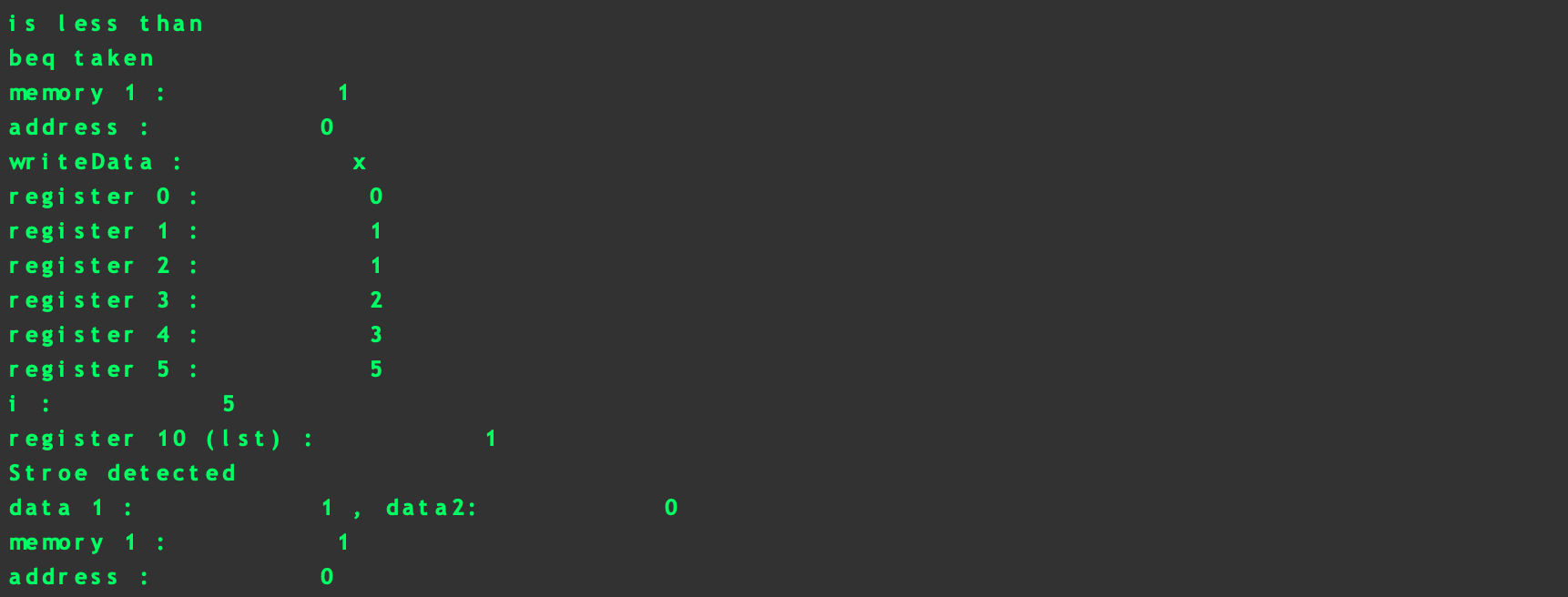
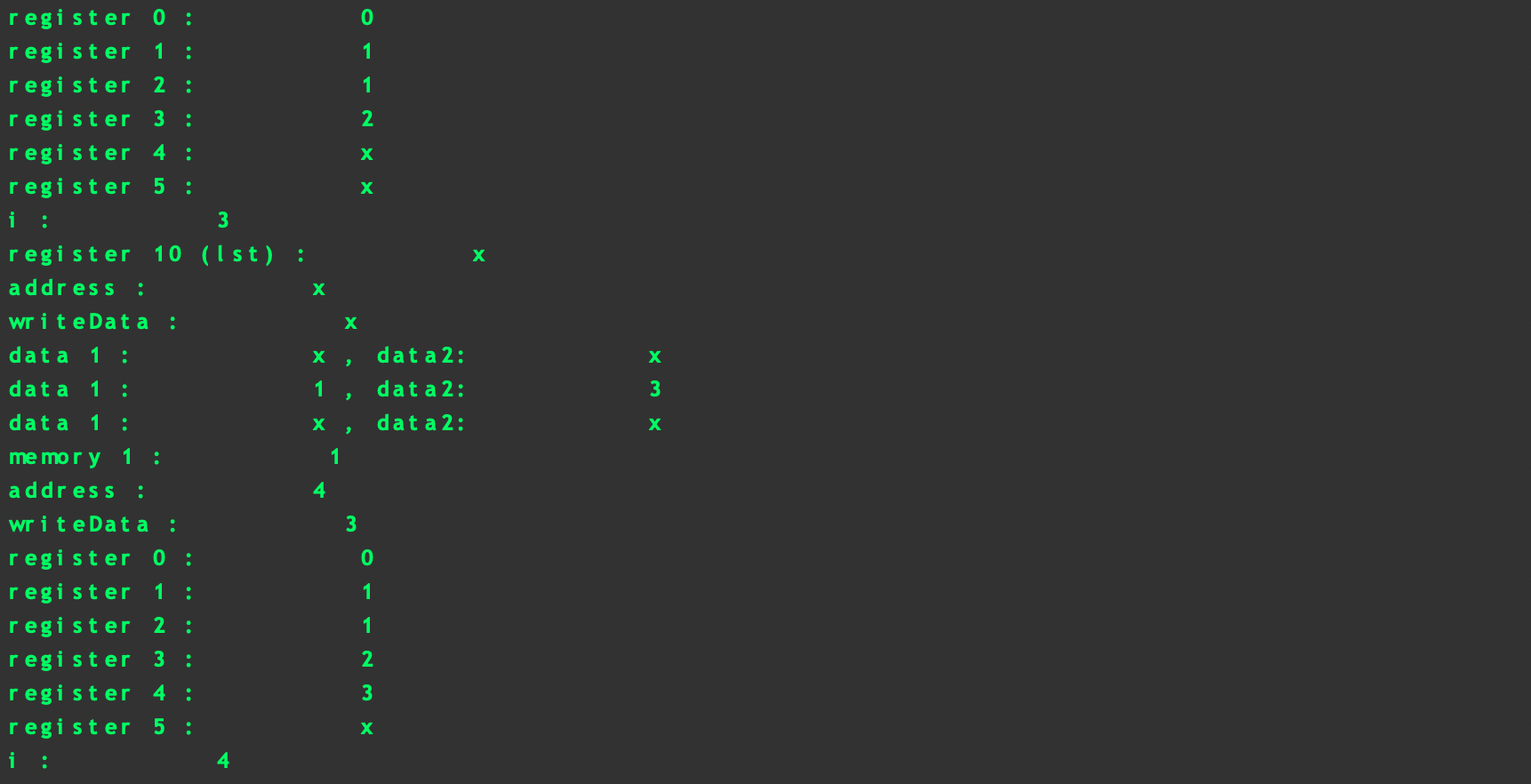
به طور کلی باید تمامی هازارد های موجود در پایپ لاین باید رفع شوند برای همین منظور دو واحد هازارد و فورواردینگ درون پردازنده قرار داده شده اند تا سیگنال های متناسب با دستور و نوع هازارد ( ۱ یا ۲ ) را ایجاد کند و تمامی ورودی های ماکس های مختلف از جمله قبل از ای ال یو را کنترل کند. دو نوع هازارد را با بخش زیر کنترل میکنیم : 

و همچنین در قسمت فورواردینگ داریم :‌



به طور کلی پروژه شامل قسمت های مختلفی دیگر از جمله رجیستر های بین دو استیج و .... می‌باشد . حال برای تست برنامه فیبوناچی ما در ابتدا دو دستور لود را امتحان میکنیم بدین شکل که در مموری دو مقدار پایه فیبوناچی ۱ و ۰ را قرار می‌دهیم و در ادامه با دستور اد مقادیر بالاتر برنامه را حساب میکنیم و درنهایت با رسیدن به پنجمین جمله فیبوناچی برنچ میکنیم . برای تست دستور sw هم حاصل نتیجه مقایسه دو رجیستر را درون یک رجیستر سوم ریخته و مقدار آن را درون مموری ذخیره می‌کنیم .

که خروجی برنامه و طول موج های آن به شکل زیر می‌باشد :  
  




بخش ب پروژه بدین صورت می باشد که باید دستور ذخیره در حافظه را در طول ۲ کلاک انجام بدیم . برای بخش ب پروژه با توجه به خواسته سوال از این روش استفاده شده که در کلاک اول ۱۶ بیت کم ارزش و در کلاک دوم ۱۶ بیت پرارزش دیتا را درون مموری سیو کنیم . در استفاده از کامپوننت های قبلی تغییری ایجاد نمیکنیم و خواسته سوال را در بخش دیتا مموری هندل میکنیم اما چون با این کار ماهیت پایپ لاین به ۶ استیج تغییر می‌یابد نیاز به تعریف یک سری از رجیستر ها سنکرون با کلاک برای انتقال دیتا از استیج قبلی به استیج بعدی می‌باشند .

برای این کار دیتا مموری به شکل زیر در می آید :  
  