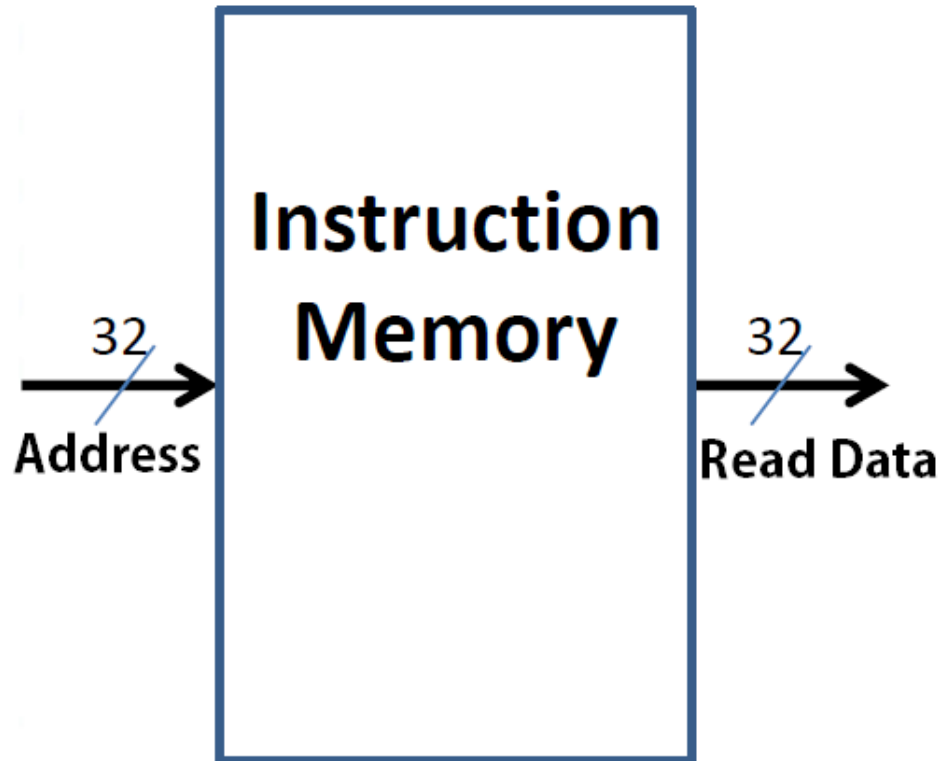


قسمت اول – پیاده سازی Instruction Memory Module

هدف در این قسمت، پیاده سازی ماژول Instruction Memory است.



ماژول مذکور باید شامل یک ورودی باشد:

ورودی 32 بیتی برای آدرس PC (Address)

ماژول مذکور باید شامل یک خروجی باشد:

خروجی 32 بیتی مربوط به دستور موجود در رجیستر با آدرس PC موجود در Instruction Memory (Read Data)

نکته: به دلیل نبود حافظه فیزیکی برای Instruction Memory، حافظه مذکور در یک فایل با نام memfile.dat در اختیار شما قرار گرفته است.

توضیحات پیرامون فایل memfile.dat:

این فایل شامل تعدادی عدد به فرمت HEX در هر خط می باشد که معادل با دستور های 32 بیتی مذکور برای پردازنده ما می باشند.

نحوه خواندن داده ها از فایل memfile.dat:

برای خواندن داده های موجود در فایل memfile.dat و ذخیره آن ها در یک رجیستر برای ارتباط با ماژول Instruction Memory همانند زیر عمل می کنیم:

ابتدا یک رجیستر 2 بعدی برای حافظه مورد نظر در ماژول خود به شکل زیر تعریف کنید:

```
reg [31:0] RAM [1023:0];
```

به این معنا که حافظه RAM مذکور شامل 1024 خانه حافظه که هر کدام محتوی 32 بیت رجیستر هستند می باشد. (یک حافظه 4 کیلو بایتی)

سپس به کمک دستور زیر:

initial

```
$readmemh ("memfile.dat", RAM);
```

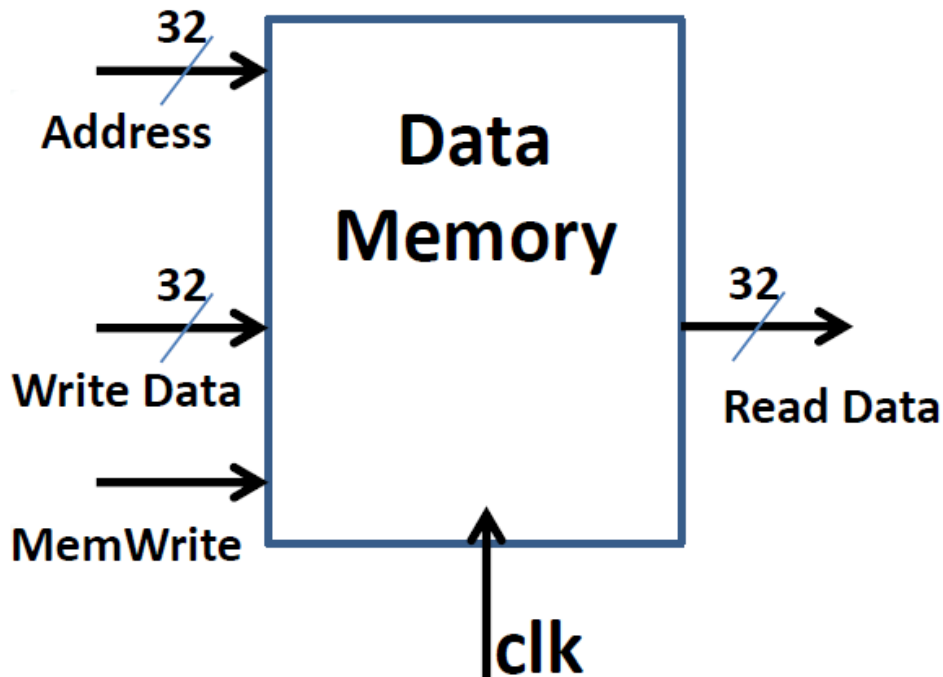
محتوای موجود در فایل memfile.dat در رجیستر 2 بعدی RAM به این صورت قرار می گیرد که از اولین خط از فایل مربوطه تا آخرین خط (خط n ام)، داده های HEX به ترتیب در رجیستر های RAM[0] تا RAM[n] که هر کدام 32 بیت هستند قرار می گیرند.

هدف module:

هدف طراحی ماژولی با ورودی و خروجی های ذکر شده و حافظه RAM تعریف شده به شکل بالا است که با دریافت یک آدرس 32 بیتی به عنوان ورودی (Address)، رجیستر 32 بیتی خروجی را با مقدار RAM[Address] مقدار دهی کند.

قسمت دوم – پیاده سازی Data Memory Module

هدف در این قسمت، پیاده سازی ماژول Data Memory است.



ماژول مذکور باید شامل چهار ورودی باشد:

1. ورودی یک بیتی برای clk (clk)
2. ورودی یک بیتی برای بیت کنترلی MemWrite حافظه (MemWrite)
3. ورودی 32 بیتی برای آدرس خانه حافظه (Address)
4. ورودی 32 بیتی مربوط به داده مورد نظر برای ذخیره در رجیستر با آدرس ورودی موجود در Data Memory (Write Data)

ماژول مذکور باید شامل یک خروجی باشد:

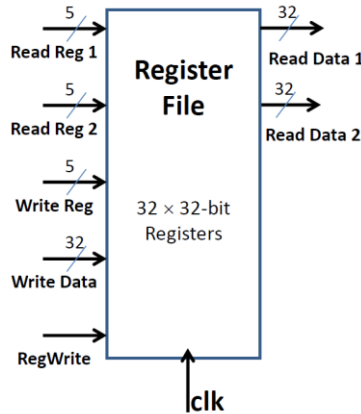
خروجی 32 بیتی مربوط به داده موجود در رجیستر با آدرس ورودی موجود در Data Memory (Read Data)

هدف module:

هدف طراحی ماژولی با ورودی و خروجی های ذکر شده و یک حافظه RAM دو بعدی با 1024 خانه 32 بیتی می باشد که با دریافت یک آدرس 32 بیتی word aligned به عنوان ورودی (Address)، همواره مقدار موجود در خانه Address را برای خواندن در رجیستر خروجی قرار می دهد. (مدار ترکیبی و بدون نیاز به clk) و هم چنین، در صورت فعال بودن بیت MemWrite، در انتهای کلاک بعد مقدار موجود در ورودی Write Data را در خانه حافظه RAM با آدرس 32 بیتی Address قرار می دهد.

قسمت سوم – پیاده سازی Register File Module

هدف در این قسمت، پیاده سازی ماژول Register File است.



ماژول مذکور باید شامل شش ورودی باشد:

1. ورودی یک بیتی برای clk (clk)
2. ورودی یک بیتی برای بیت کنترلی Write Enable رجیستر فایل (RegWrite)
3. ورودی پنج بیتی برای اشاره به آدرس رجیستر اول (Read Reg 1)
4. ورودی پنج بیتی برای اشاره به آدرس رجیستر دوم (Read Reg 2)
5. ورودی پنج بیتی برای اشاره به آدرس رجیستر سوم (Write Reg)
6. ورودی 32 بیتی مربوط به داده مورد نظر برای ذخیره در رجیستر با آدرس Write Reg موجود در رجیستر فایل. (Write Data)

ماژول مذکور باید شامل دو خروجی باشد:

1. خروجی 32 بیتی مربوط به داده موجود در رجیستر با آدرس Read Reg 1 موجود در رجیستر فایل. (Read Data 1)
2. خروجی 32 بیتی مربوط به داده موجود در رجیستر با آدرس Read Reg 2 موجود در رجیستر فایل. (Read Data 2)

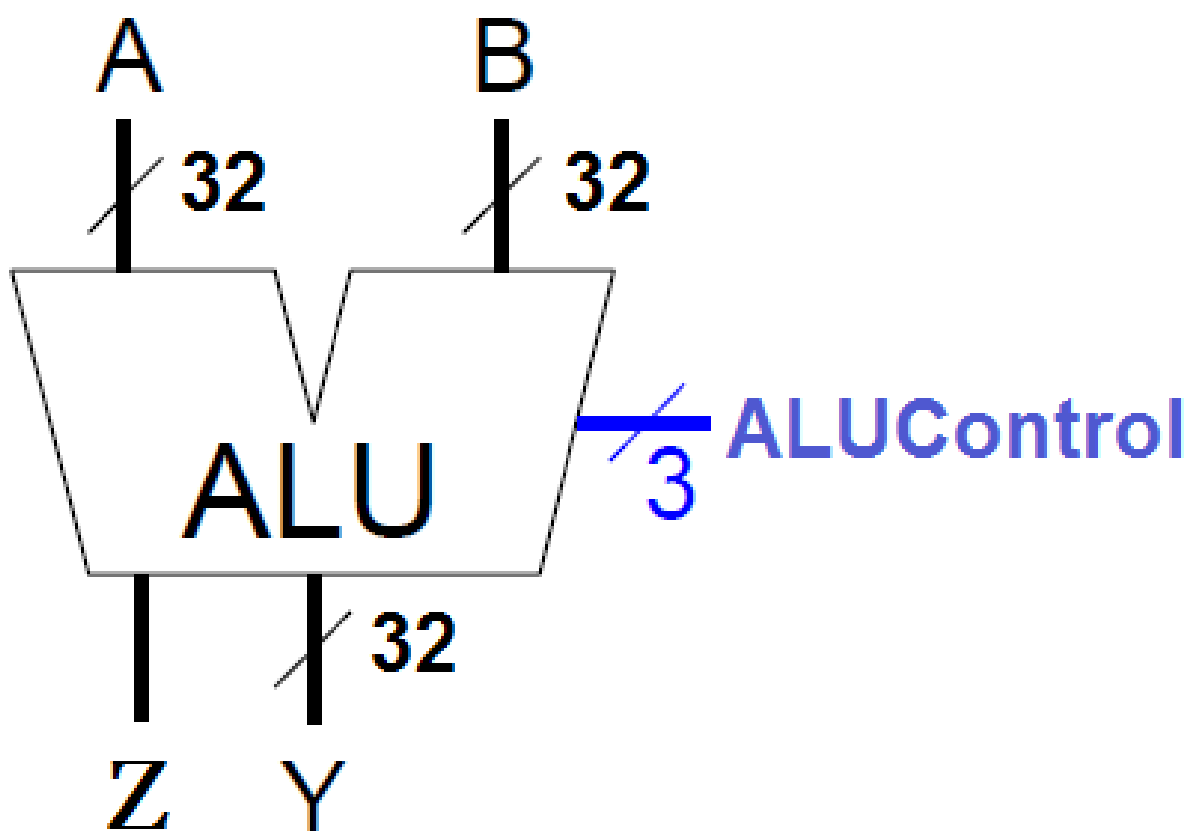
هدف module:

هدف طراحی ماژولی با ورودی و خروجی های ذکر شده و یک رجیستر فایل دو بعدی با 32 رجیستر 32 بیتی می باشد که با دریافت سه آدرس 5 بیتی به عنوان ورودی، همواره رجیستر ها با آدرس های Read Reg 1 و Read Reg 2 را در خروجی های Read Data 1 و Read Data 2 قرار می دهد. (مدار ترکیبی و بدون نیاز به clk)

و هم چنین، در صورت فعال بودن بیت RegWrite، در انتهای کلاک بعد مقدار موجود در ورودی Write Data را در رجیستر با آدرس 5 بیتی Write Reg قرار می دهد.

قسمت چهارم – پیاده سازی ALU Module

هدف در این قسمت، پیاده سازی ماژول ALU است.



ماژول مذکور باید شامل سه ورودی باشد:

1. ورودی اول 32 بیتی. (A)
2. ورودی دوم 32 بیتی. (B)
3. ورودی 3 بیتی کنترلی ALU. (ALUControl)

ماژول مذکور باید شامل دو خروجی باشد:

1. خروجی 32 بیتی برای نتیجه ALU. (Y)
2. خروجی یک بیتی برای نشان دادن صفر بودن خروجی ALU. (Z)

هدف module:

هدف طراحی ماژولی با ورودی و خروجی های ذکر شده می باشد که با دریافت دو رجیستر 32 بیتی به عنوان ورودی، با توجه به ALUControl همواره عملیات مورد نظر را که در زیر آمده اند بر روی دو رجیستر انجام داده و نتیجه خروجی را در رجیستر با نام Y قرار می دهد.

نکته: در صورت صفر شدن Y، خروجی یک بیتی Z فعال می شود.

جدول عملیات های ALU با توجه به 3 بیت ALUControl:

Operation	ALUControl[2:0]
ADD	3'b000
SUB	3'b001
AND	3'b010
OR	3'b011
XOR	3'b100
NOR	3'b101
SLT	3'b110
SLTU	3'b111

خواسته های پروژه :

- 1- طراحی کد وریلاگ واحد های بالا و کارکرد صحیح این مدار با تست بنچی که در اختیارتان قرار داده شده است یا تست بنچی که خودتان زده اید
- 2- گزارشی کوتاه در مورد نحوه نوشتن واحد ها

نکات مهم پروژه :

- 1- زمان تحویل حضوری در CW اعلام می شود .
- 2- هر واحد خود را در فولدري جداگانه و به همراه تست بنچ آن قرار دهید و در غالب فایلی با اسم Modules_Student#.rar در CW آپلود کنید .