**به نام خدا**

گزارشکار فاز اول پروژه درس معماری کامپیوتر

**طراحی و پیاده­سازی Data Path**

استاد

**دکتر حمید سربازی آزاد**

اعضای گروه

**محمدپارسا بشری 400104812**

**محسن قاسمی 400105166**

**امیرحسین رازلیقی 99102423**

بهار 1402

فهرست

[مقدمه و هدف فاز اول 2](#_Toc136021513)

[طراحی Register File 2](#_Toc136021514)

[طراحی Memory 2](#_Toc136021515)

[طراحی ALU 3](#_Toc136021516)

[طراحی منطق Program Counter 4](#_Toc136021517)

[طراحی Control Unit 5](#_Toc136021518)

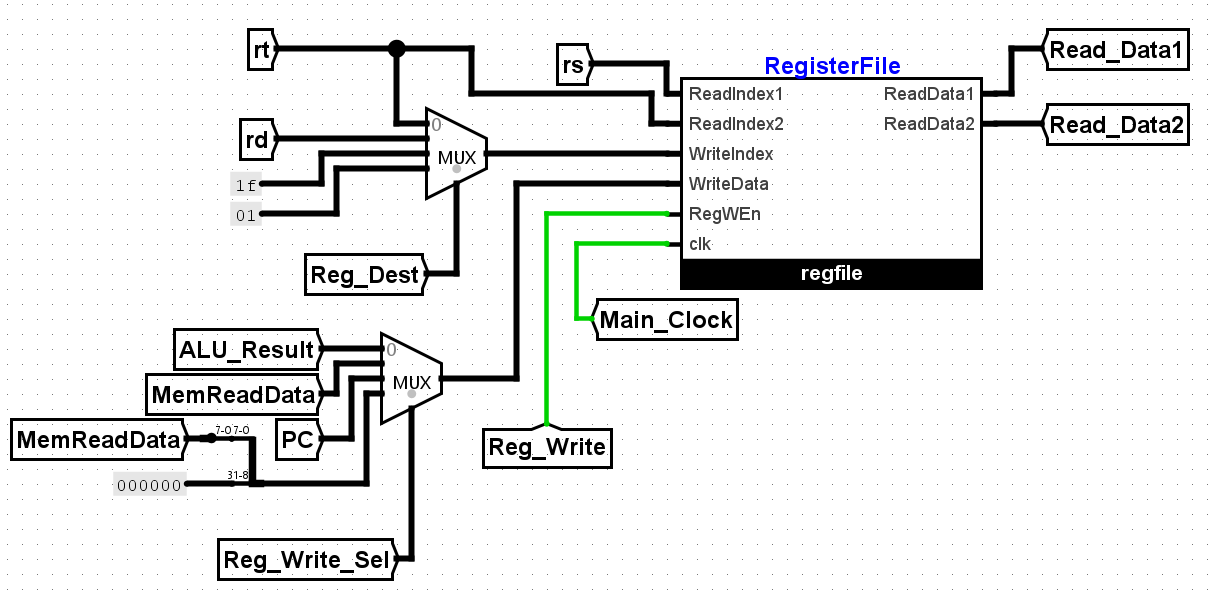
[طراحی ALU Control 6](#_Toc136021519)

[تست اجزای مدار 7](#_Toc136021520)

# مقدمه و هدف فاز اول

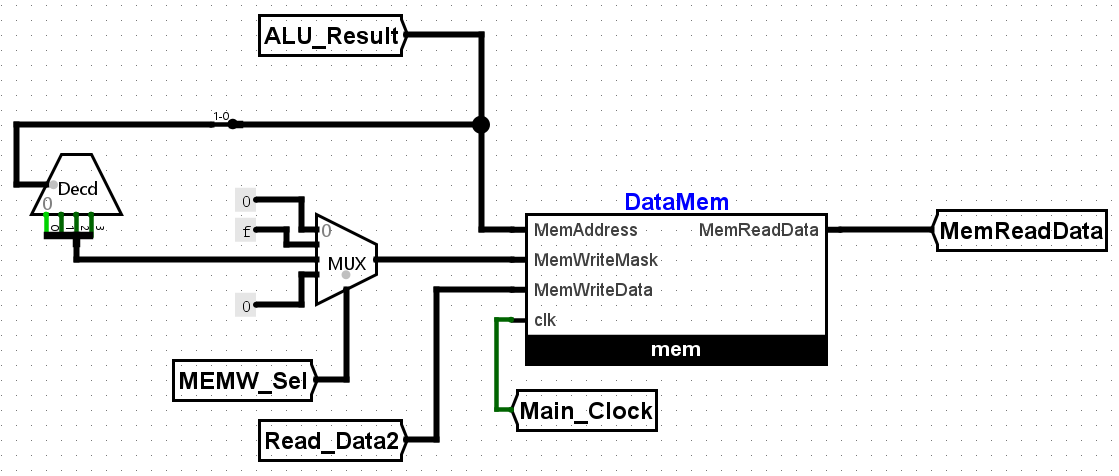
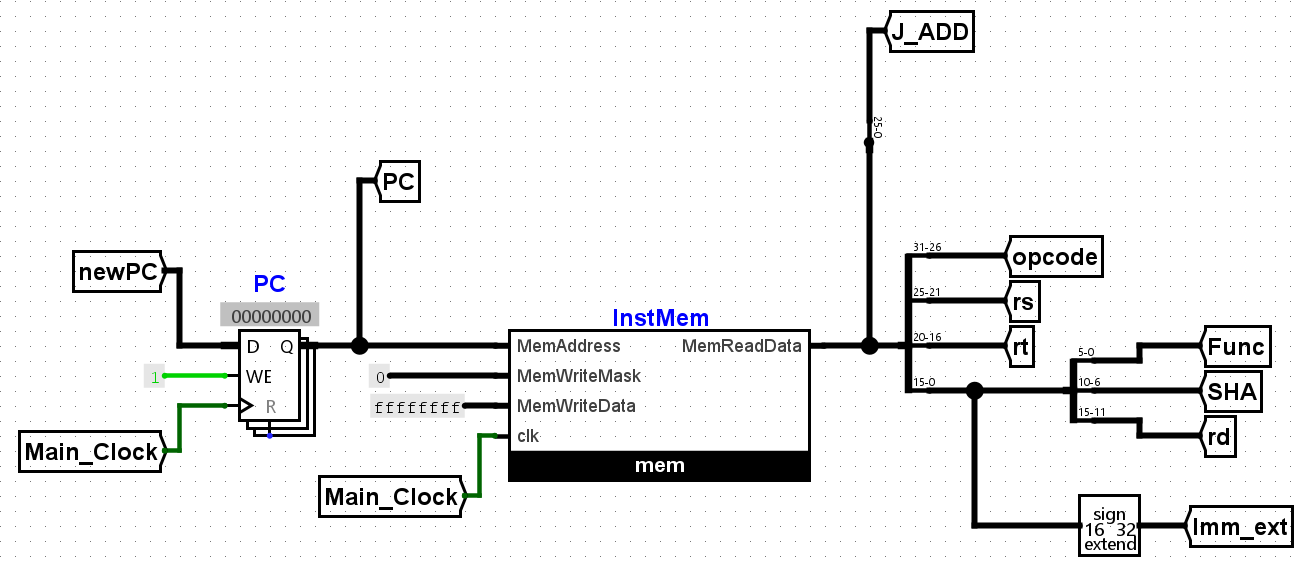
هدف کلی این پروژه، طراحی و پیاده­سازی یک پردازنده MIPS است. در فاز اول قصد داریم Datapath و Control Unit این پردازنده را به صورت Single Cycle طراحی و پیاده­سازی کنیم. همچنین برای اطمینان از صحت عملکرد Component های پردازنده به صورت خودکار، تعدادی تست نیز برای ماژول­هایمان می­نویسیم. ماژول­های Register File و Memory از قبل در اختیارمان قرار گرفته بود، بنابراین به توضیح مختصری درباره آن­ها اکتفا می­کنیم.

# طراحی Register File

پردازنده MIPS پردازنده­ای 32 بیتی با معماری RISC است که 32 عدد ثبات عمومی دارد. طراحی Register File به سادگی شامل 32 عدد ثبات می­شود که ورودی Write Enable آنها با استفاده از یک دیکودر 5×32 تولید می­شود (ثبات شماره 0 قابل نوشتن نیست و همواره مقدار 0 را نگه می­دارد). دو ورودی 5 بیتی ReadIndex1 و ReadIndex2 نیز به عنوان ورودی select به دو MUX 5×32×32 داده می­شوند تا محتویات ثبات­هایی که باید خوانده شوند را انتخاب کنند.

شکل 1- ورودی­ها و خروجی­های Register File

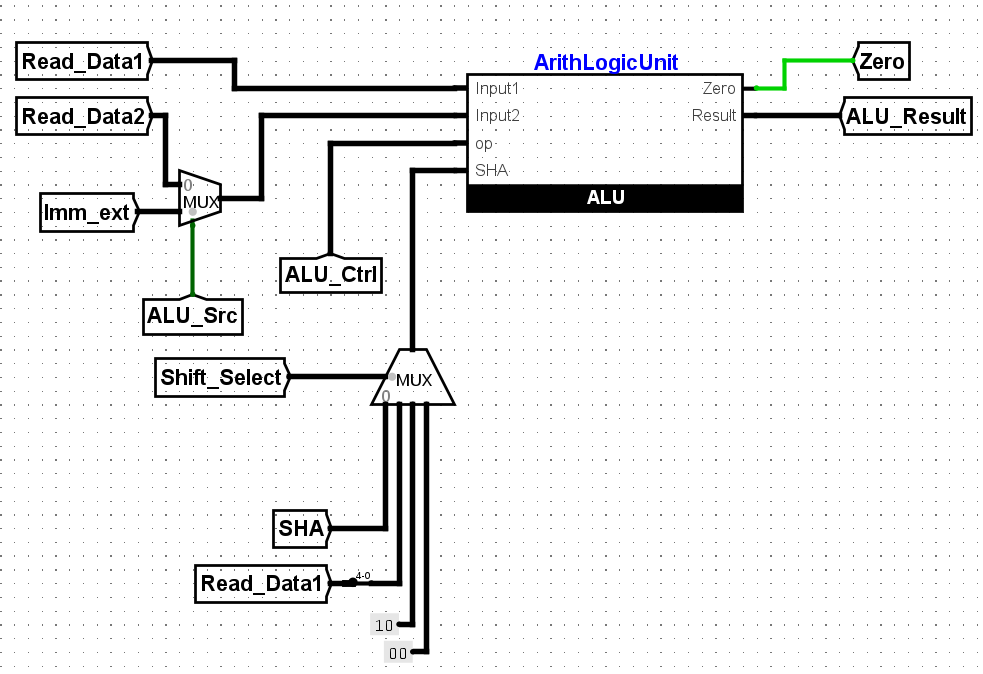
# طراحی Memory

از دو حافظه مجزا برای داده­ها و دستورات استفاده می­کنیم (Data Memory و Instruction Memory) که هر کدام شامل 16K کلمه هستند و از آنجایی که هر کلمه 4 بایت است، پس هر ماژول حافظه 64KB ظرفیت دارد که با استفاده از 4 عدد RAM 16K×8 و ساختاری شبیه low-order interleaving ساخته می­شود. به این صورت که دو بیت سمت راست را از Address حذف می­کنیم و 14 بیت باقی­مانده را به هر چهار ماژول می­دهیم. سپس برای نوشتن یک (یا چند) بایت خاص داخل یک کلمه، از ورودی Mask استفاده می­کنیم.

شکل 2- ورودی­ها و خروجی­های Instruction Memroy

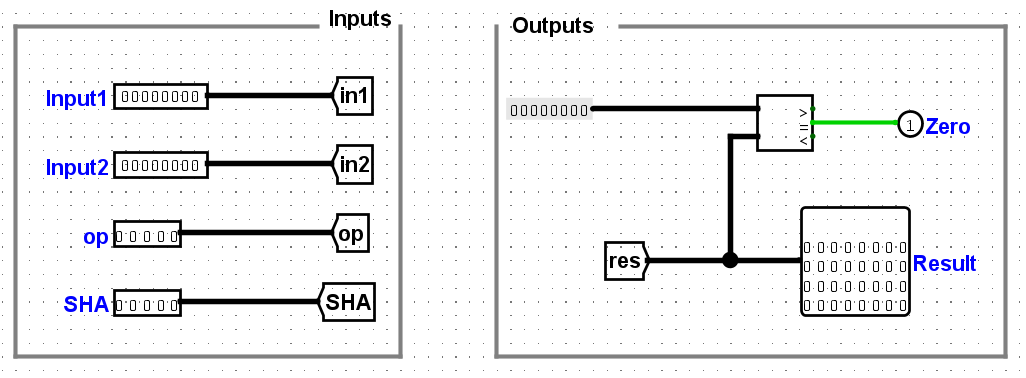
شکل 3- ورودی­ها و خروجی­های Data Memory

# طراحی ALU

واحد محاسبات و منطق (ALU) وظیفه انجام عملیات­های اصلی روی داده­ها را دارد. این واحد دو ورودی دیتا (32 بیتی) دریافت می­کند و با توجه به ورودی operation (5 بیتی)، عملیاتی که باید روی این دو داده انجام شود را انتخاب می­کند. سپس خروجی عملیات انجام شده را از طریق خروجی Result (32 بیتی) اعلام می­کند. همچنین خروجی Zero (1 بیتی) صفر بودن حاصل محاسبه را نشان می­دهد.

شکل 4- ورودی­ها و خروجی­های ALU

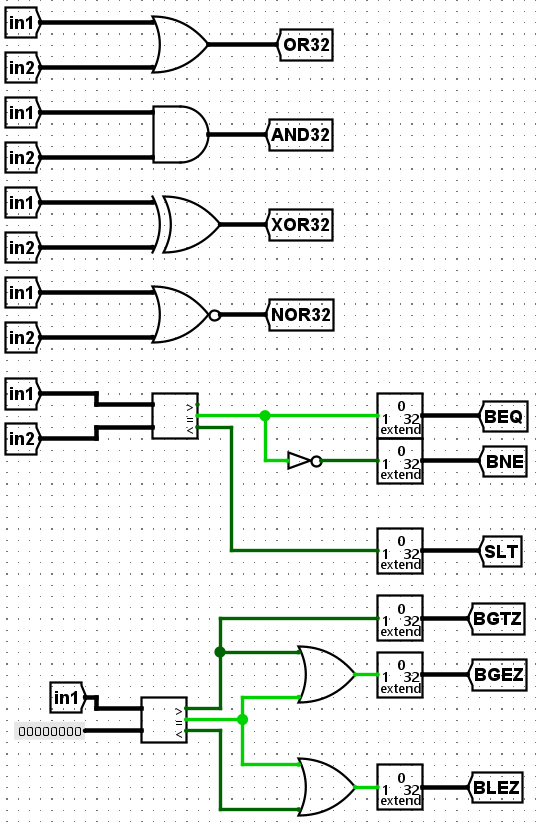
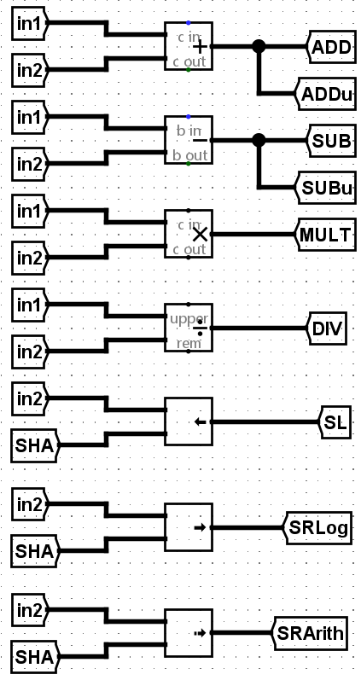
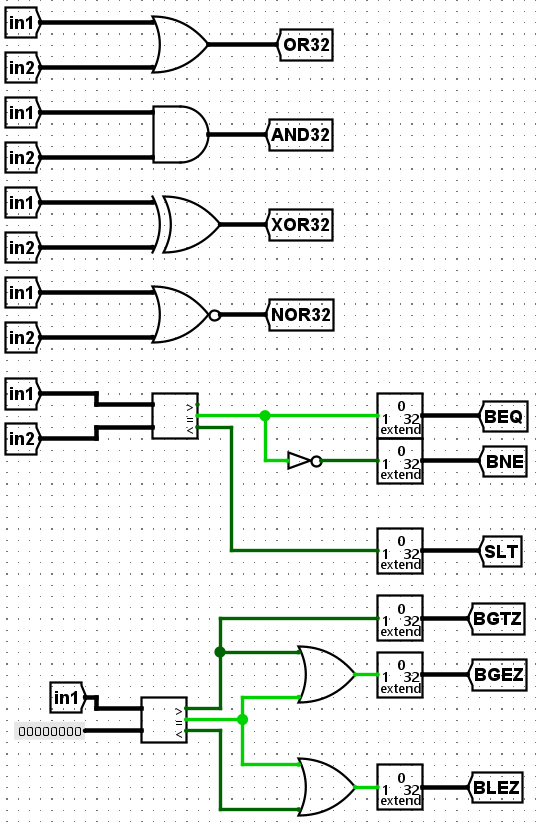
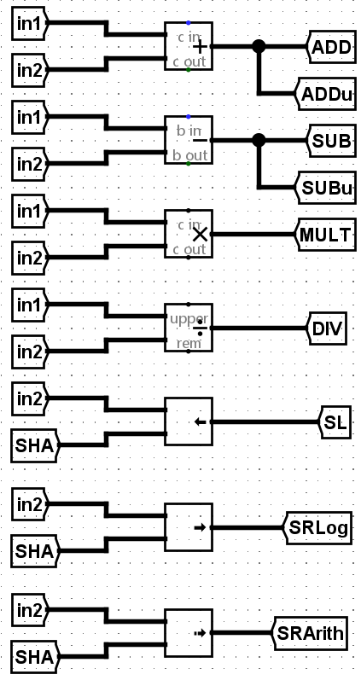
همچنین ورودی SHA مقدار شیفت را مشخص می­کند. توجه کنید که ورودی­های ALU توسط CU تولید می­شوند.

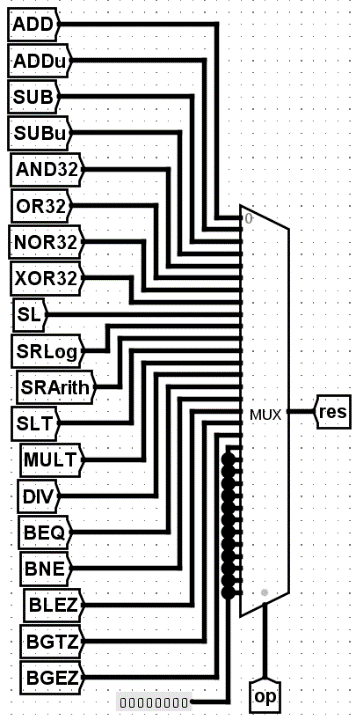
حالا به طراحی داخلی ALU می­پردازیم. ابتدا ترمینال­های ورودی و خروجی را می­سازیم. توجه کنید که خروجی Zero صرفا چک می­کند که آیا Result صفر است یا خیر.

شکل 5- ترمینال­های ورودی و خروجی ALU

بدنه اصلی ALU شامل کامپوننت­هایی­ است که محاسبات را انجام می­دهند. در این قسمت تمام خروجی­های ممکن را تولید می­کنیم.

شکل 6- تولید تمام خروجی­های ممکن در ALU



سپس با استفاده از یک MUX 5×32×32 و با توجه به ورودی operation خروجی مورد نظرمان را انتخاب می­کنیم.

شکل 7- انتخاب خروجی مورد نظر با توجه به بیت­های op

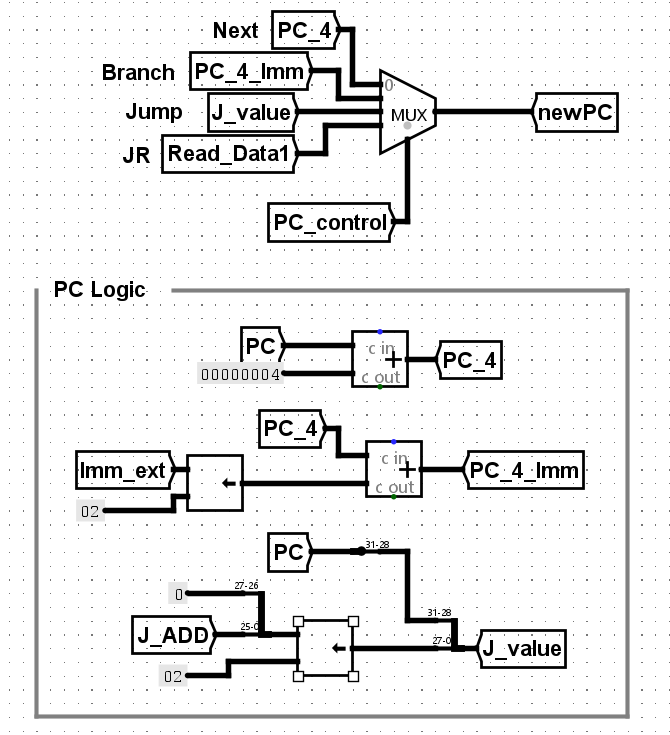
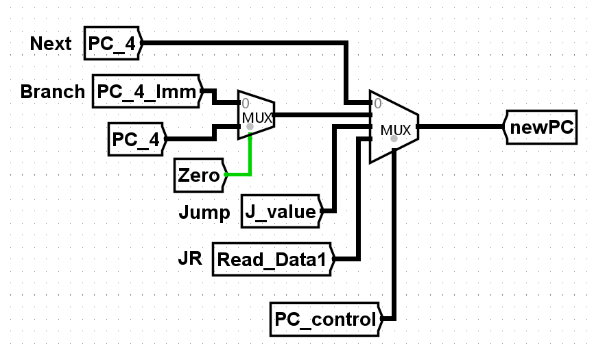
به ورودی­های خالی MUX هم عدد 0 را وصل می­کنیم. دلیل این کار را در قسمت طراحی ALU Control متوجه خواهیم شد.

# طراحی منطق Program Counter

|  |  |
| --- | --- |
| Case | Next PC |
| Normal execution (next instruction) | PC + 4 |
| Branch instructions in I-format | PC + 4 + SIGN\_EXTEND(Imm|00) |
| Jump instructions in J-format (J and JAL) | PC [31:28] | Address | 00 |
| JR (jump register) instruction in R-format | $rs |

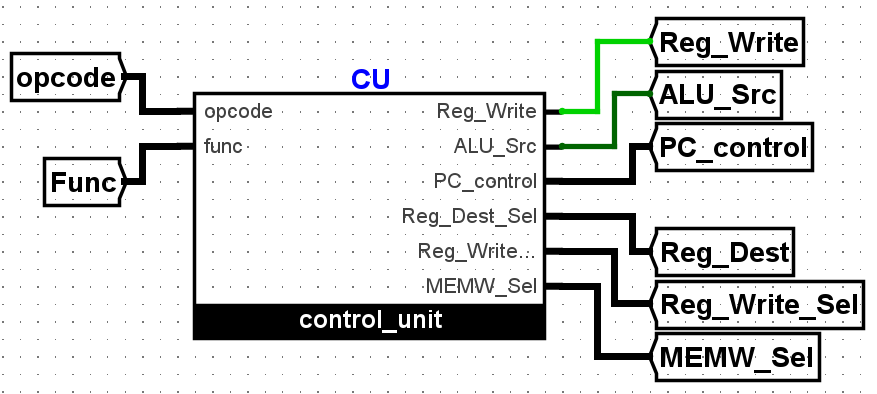
همانطور که در شکل 2 مشاهده کردیم، ثبات Program Counter یا به اختصار PC، یک ثبات 32 بیتی است که آدرس دستور بعدی را در خود نگه می­دارد. با توجه به طراحی Single Cycle در این فاز، PC باید با هر کلاک تغییر کند. مقدار جدید PC باید از بین چهار مقدار زیر انتخاب شود:

جدول 1- مقادیر جدید PC و روش انتخاب از بین آنها

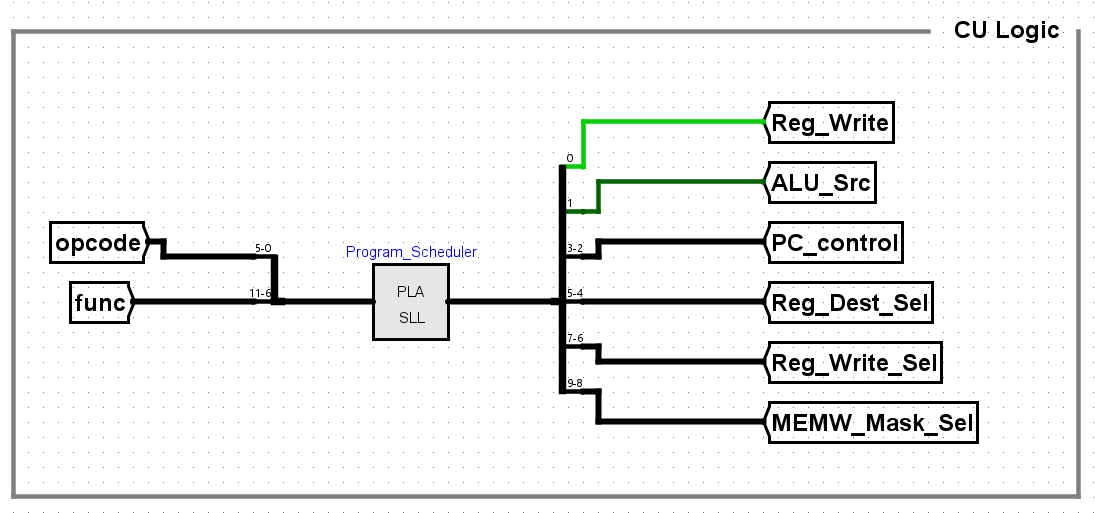
برای پیاده­سازی منطق این کار، ابتدا چهار مقدار بالا را می­سازیم و سپس با استفاده از یک MUX 2×4×1 از بین این چهار مقدار یکی را انتخاب کرده و به ورودی PC وصل می­کنیم تا با کلاک بعدی وارد PC شود. توجه کنید که ورودی select این مالتی پلکسر از CU می­آید.

شکل 8- طراحی منطق PC

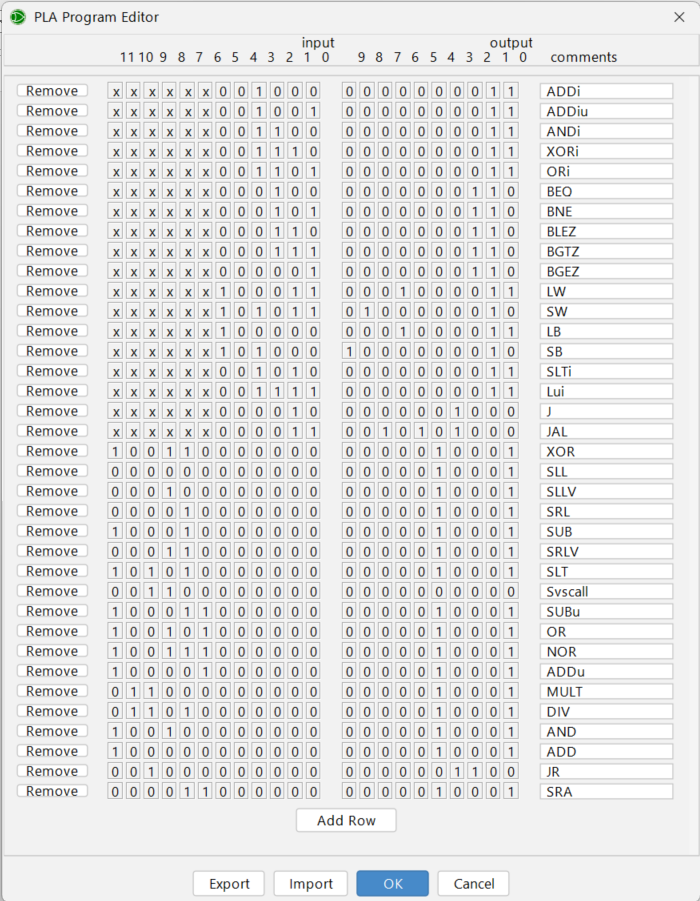
# طراحی Control Unit

وظیفه واحد کنترل یا CU این است که با گرفتن Opcode و Func (هر دو 6 بیتی) که از قالب دستور می­آیند، خروجی­های کنترلی لازم برای بقیه مدار (Datapath) را تولید کند. خروجی­هایی که باید تولید شوند به مدارهایی که در قسمت­های قبل آمده­اند، داده می­شوند.

شکل 9- ورودی­ها و خروجی­های Control Unit

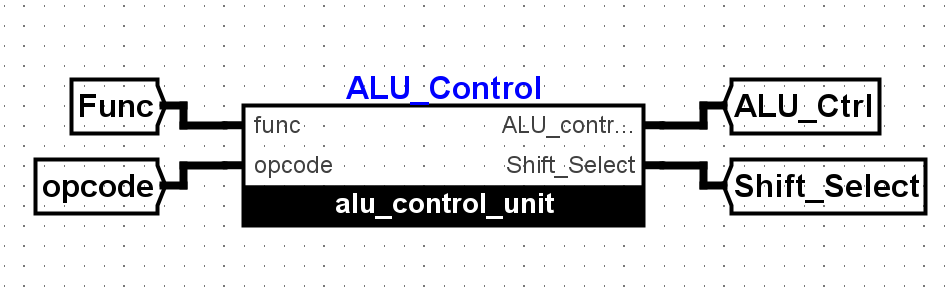
طراحی داخلی CU را با استفاده از یک PLA انجام می­دهیم. پس داخل ماژول CU به شکل زیر خواهد بود.

شکل 10- استفاده از PLA در طراحی Control Unit

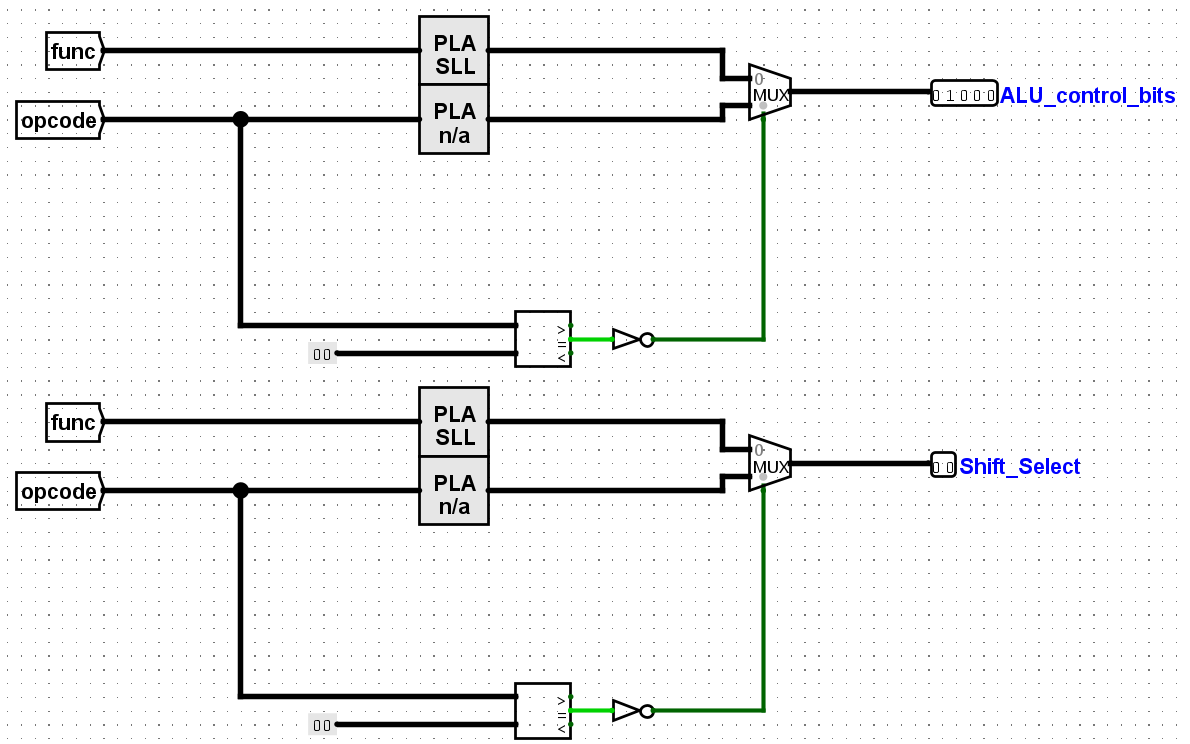
داخل PLA را با توجه به عملکرد مورد نیاز هر دستور، program می­کنیم.

شکل 11- مشخص کردن منطق داخل PLA

# طراحی ALU Control

عملکرد ALU Control هم مانند CU است؛ یعنی به عنوان ورودی Opcode و Func را گرفته و خروجی­های کنترلی مربوط به ALU را تولید می­کند.

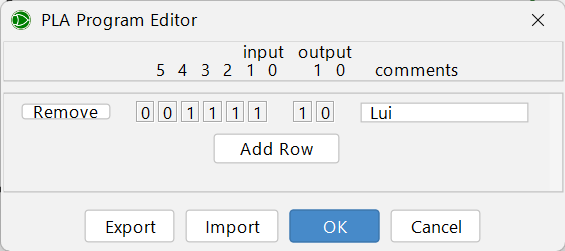
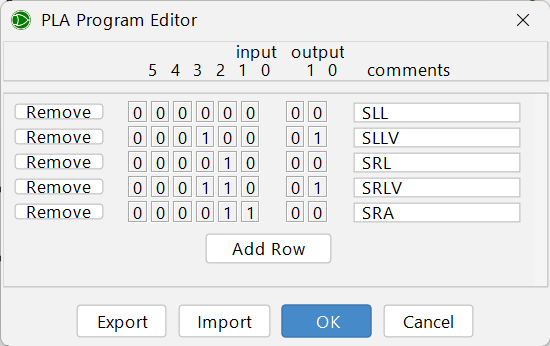
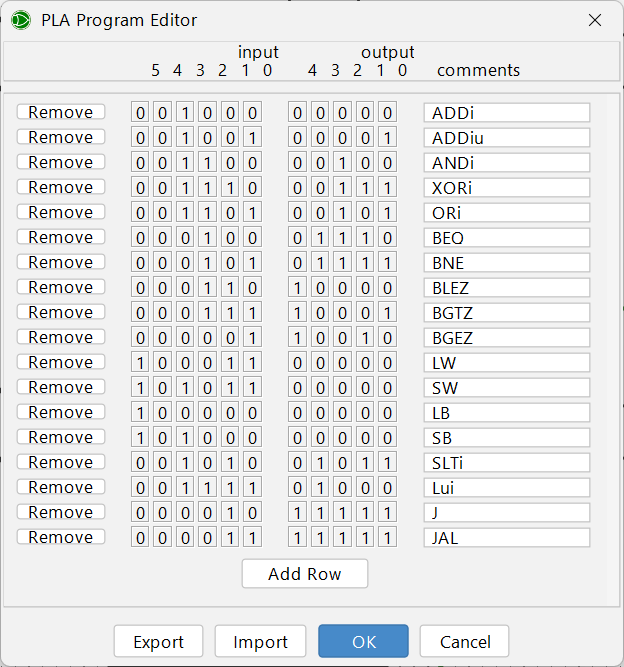
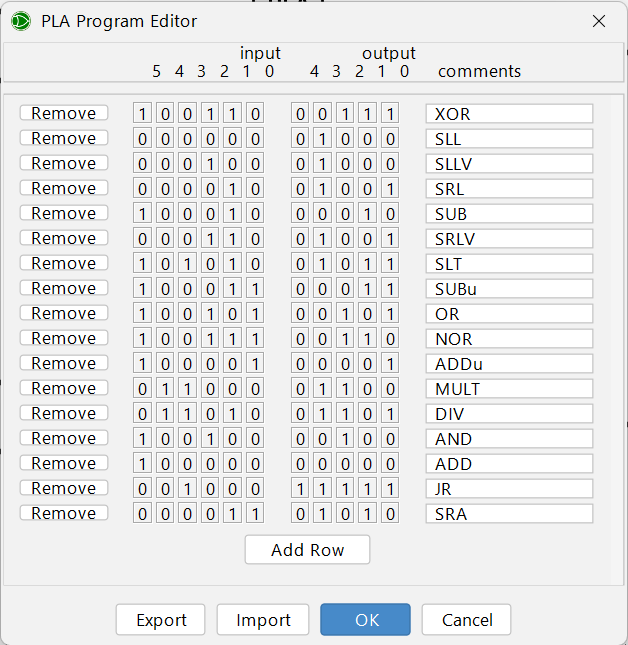
شکل 12- ورودی­ها و خروجی­های ALU Control

برای طراحی داخلی ALU Control از 4 عدد PLA استفاده می­کنیم. که به صورت زیر به خروجی­ها متصل شده­اند.

شکل 13- طراحی داخل ALU Control

سپس منطق داخل هر PLA را مشخص می­کنیم.

شکل 14- مشخص کردن منطق داخل PLA ها



# تست اجزای مدار

سلام