دانشکده مهندسی کامپیوتر

درس معماری کامپیوتر

16-bit RISC CPU

مدرس..... دکتر کاشی

تاریخ تحویل .... 1403 /11/5

آداب نامه پروژه:

- برای زدن بخش های مختلف پروژه میتوانید این بخش هارا بین خود تقسیم کنید و نیازی به اینکه حتما در زدن همه ی بخش ها تمام اعضای تیم مشارکت داشته باشند نیست. اما دقت شود در زمان ارائه پروژه، ازتمام اعضای تیم در مورد تمامی بخش ها سوال پرسیده می شود.
- این پروژه باید به زبان VHDL زده شود.
- هر بخش از پروژه حتما در یک فایل جداگانه زده شود.
- برای تمامی بخش ها باید تست نوشته شود. دقت کنید که درصورتی که برای بخشی تستی ننوشته باشید و یا آن بخش نتواند به درستی کار کند، نمره ی آن بخش را ازدست خواهید داد.
- برای تست کلی پروژه نیز یک برنامه فرضی درنظر گرفته و آن برنامه را روی CPU نوشته شده خود اجرا کنید.
- در صورت مشاهده تشابه غیرعادی بین دو گروه، نمره کل پروژه برای هر دو گروه صفر در نظر گرفته خواهد شد.

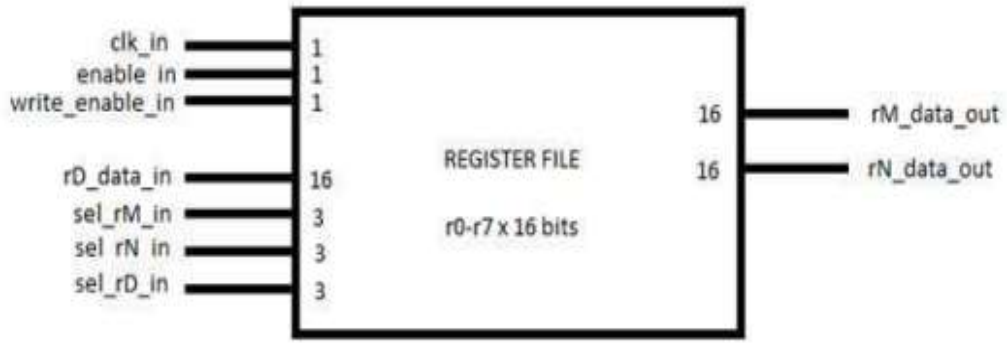
مقدمه:

در این پروژه قرار است یک معماری CPU 16 بیتی بدون Pipeline و 6 مرحله ای کامل را پیاده سازی کنید.
این CPU دارای 6 جزء است: یک register file، یک Decoder، یک ALU، یک control unit، یک PC و رم/حافظه.

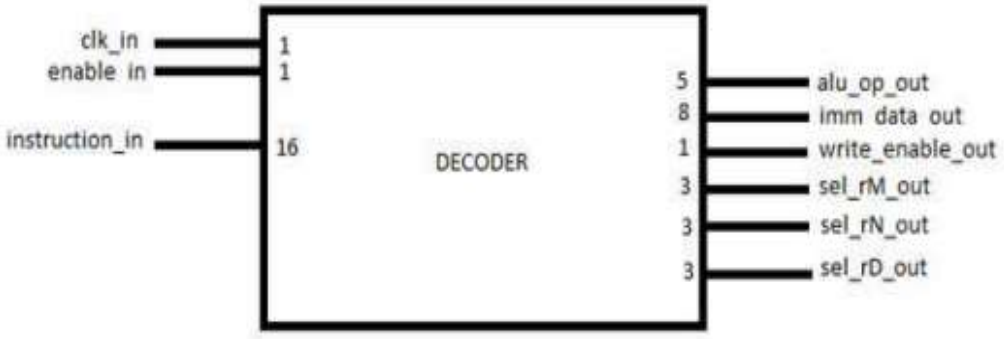
بخش های رم و register file و cpu به صورت آماده در اختیار شما قرار خواهد گرفت و وظیفه شما پیاده سازی سایر قسمت هاست.

در تصاویر زیر میتوانید نمای کلی هرکدام از بخش هارا ببینید.

Register file:

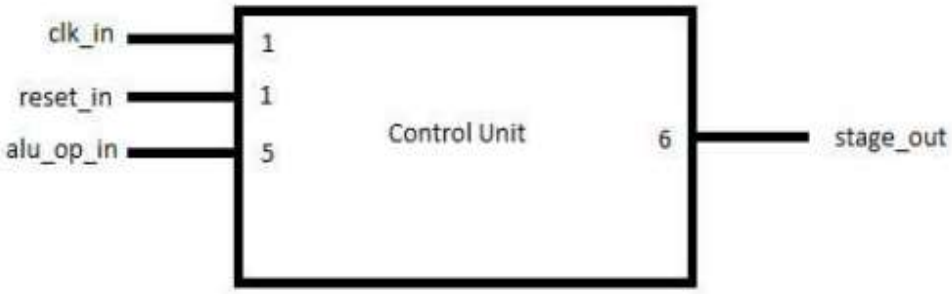


Decoder:

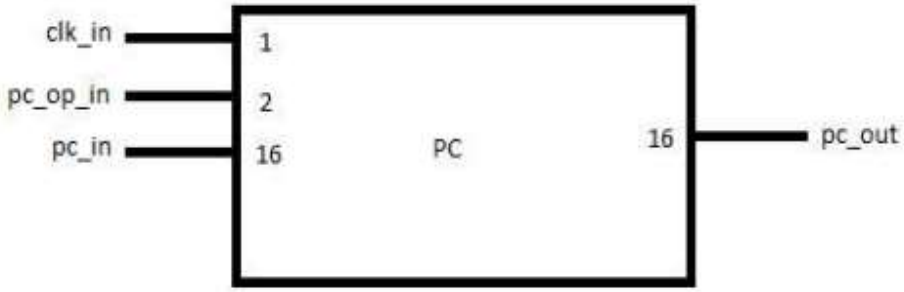


ALU:

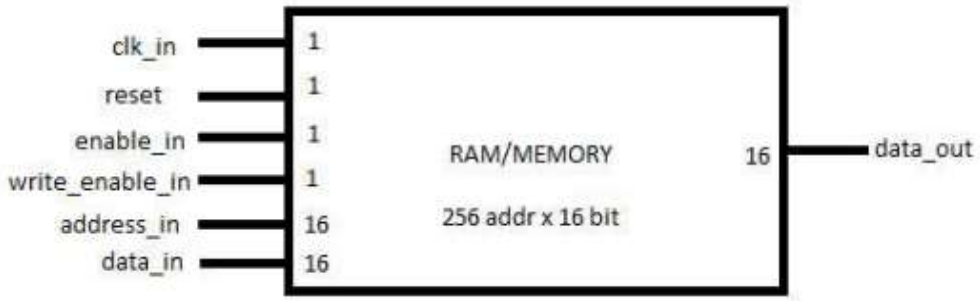
Control unit:



PC:



RAM



بررسی اجمالی:

- Opcode در این CPU برابر 4 بیت است.
- 14 دستور العمل متفاوت باید پیاده سازی شود.
- 8 رجیستر در این CPU وجود دارند.
- معماری شبیه سازی شده 512 بایت دارد.
- کلاک شبیه سازی شده دارای دوره ی 10ns یا 100Mhs است.

Instructions

```
Terms:

R = register

U = unused

rD = destination register

rM = first operand register

rN = second operand register

c = condition bit
```

I(8)/(5) = immediate data bits, 8 or 5

• Instruction	Form	Implementation	Condition bit	OPCODE
ADD	RRR	rD = rM + rN	c: 1/0 = signed/unsigned	0000
SUB	RRR	rD = rM - rN	c: 1/0 = signed/unsigned	0001
NOT	RRU	rD = not rN	c: N/A	0010
AND	RRR	rD = rM and rN	c: N/A	0011
OR	RRR	rD = rM or rN	c: N/A	0100
XOR	RRR	rD = rM xor rN	c: N/A	0101
LSL	RRI(5)	rD = rM << rN	c: N/A	0110
LSR	RRI(5)	rD = rM >> rN	c: N/A	0111
CMP	RRR	rD = cmp(rM, rN)	c: 1/0 = signed/unsigned	1000
B	UI(8)	PC = rM or 8-bit immediate	c: 1/0 = rM/8-bit immediate	1001
BEQ	URR	PC = rM conditional on rN	c: N/A	1010
IMMEDIATE	RI(8)	rD = 8-bit immediate	c: 1/0 = upper/lower 8-bits	1011
LD	RRU	rD = memory(rM)	c: N/A	1100

• Instruction	Form	Implementation	Condition bit	OPCODE
ST	URR	memory(rM) = rN	c: N/A	1101

Instruction layout

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRR	C	opcode				rD			rM			rN			U	
RRU	C	opcode				rD			rM			U				
URR	C	opcode				U			rM			rN			U	
RRI(5)	C	opcode				rD			rM			5-bit imm				
RI(8)	C	opcode				rD			8-bit immediate							
UI(8)	C	opcode				U			8-bit immediate							

بررسی جزئی تر componentهای موجود در پروژه:

PC

در واحد pc قرار است با بررسی pc_op_in که آن را در ورودی دریافت میکنیم در رابطه مقداری که قرار است در شمارنده برنامه load کنیم تصمیم بگیریم. برای هرکدام از حالات باید بانوجه به توضیح داده شده مقدار مناسب را در شمارنده برنامه قرار دهید.

```
when "00" => -- reset
when "01" => -- increment
when "10" => -- branch
when "11" => -- NOP
```

Decoder

در این واحد باید مقادیر صحیح مشخص کردن نوع دستور العمل، فعال بودن نوشتن در مموری، ثبات های مربوط به عملوند های دستورالعمل، ثبات مربوط به محل ذخیره result دستور العمل و در صورت لزوم immediate data از روی دستور العمل داده شده استخراج شود. این دستورات باید باتوجه به سیگنال های ورودی این واحد یعنی enable و clk و مقدار خود دستورالعمل گرفته شود.

Control unit

همان طور که از درس معماری کامپیوتر میدانید برای اجرای هر دستور العمل باید مراحلی مانند واکشی دستور العمل، دیکود و ... طی شوند. در این واحد باید به انجام همین کار بپردازید و اینکه کدام یک از مراحل را برای اجرای دستورالعملی که آدرس آن از pc گرفته شده است باید اجرا شود را مشخص کنید. این دستور العمل اگر خوانده شده باشد opcode مربوط به آن به عنوان ورودی به این واحد داده خواهد شد تا در واحد Execute بتوانید در رابطه با stage تصمیم بگیرید. ؛ از دیگر ورودی های این واحد reset (برای reset کردن stage(مرحله) اجرای دستور العمل) و clk است. در نهایت در خروجی باید stage و مرحله ای که درآن هستیم قرارداده شود.

```
when "000001" => -- Fetch
when "000010" => -- Decode
when "000100" => -- Reg read
when "001000" => -- Execute
when "010000" => -- Memory
when "100000" => -- Reg write
```

ALU

در نهایت در این واحد نیز باید تمام دستور العمل های گفته شده پیاده سازی کنید. نکات مربوط به overflow و اعداد علامت دار و بیت مربوط به شرط را حتما در پیاده سازی خود لحاظ کنید.