تمرین پنجم درس معماری کامپیوتر
بخش عملی

استاد: دکتر اسدی

کیان قاسمی ۴۰۱۱۰۲۲۶۴
اشکان تاریوردی ۴۰۱۱۰۵۷۵۳

بخش کنترلی

۱ ALU Control Unit

برای طراحی ALU Control Unit، ابتدا جدول مقادیر ALU Control Input به ازای $Opcode$ها و $Func$ ها رسم می‌کنیم. سپس تابع منطقی هر بیت را جداگانه با استفاده از جدول کارنو بدست آورده و در نهایت شماتیک مدار را در نرم‌افزار Quartus II رسم می‌کنیم.

Instruction opcode	ALUOp	Func field	Desired Action	ALU control input
LB	00	XXX	Add	0011
SB	00	XXX	Add	0011
BEQ	01	XXX	EQU	0110
BNQ	11	XXX	NEQU	0111
ADD	10	000	Add	0011
SUB	10	001	Sub	0100
AND	10	010	And	0000
OR	10	011	Or	0001
MULT	10	100	Mul	0101
XOR	10	101	Xor	0010
ADDI	10	000	Add	0011
SUBI	10	001	Sub	0100
ANDI	10	010	And	0000
ORI	10	011	or	0001

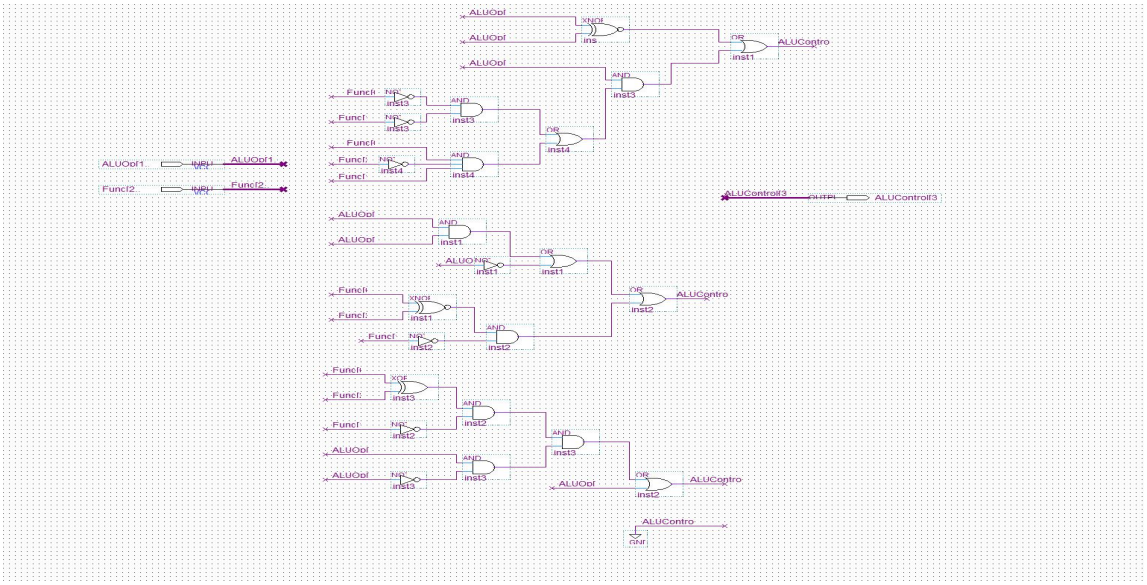
$$ALUControl[0] = OR(XNOR(ALUOp1, AlUOp0), AND(ALUOp1, OR(AND(\ Func2, AND(Func0, Func1)), AND(\ Func0,\ Func1)))$$

$$ALUControl[1] = OR(OR(\ ALUOp1, AND(AlUOp0, ALUOp1)), AND(\ Func1, XNOR(Func0, Func2)))$$

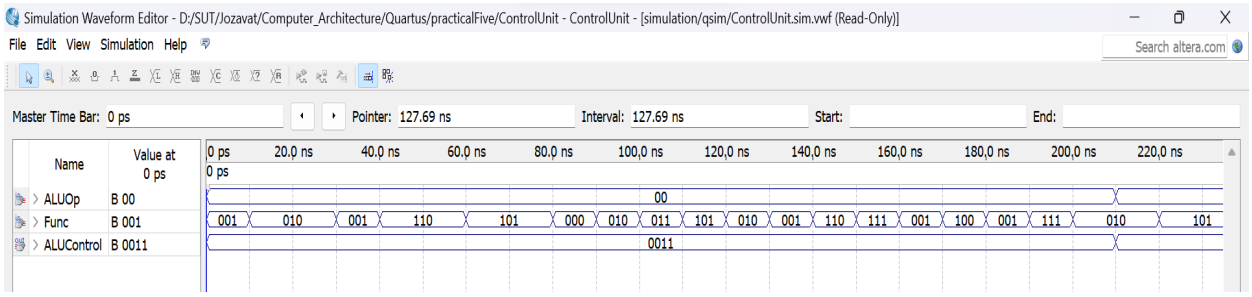
$$ALUControl[2] = OR(ALUOp0, AND(AND(\ ALUOp0, ALUOp1), AND(\ Func1, XOR(Func0, Func2))))))$$

$$ALUControl[3] = 0$$

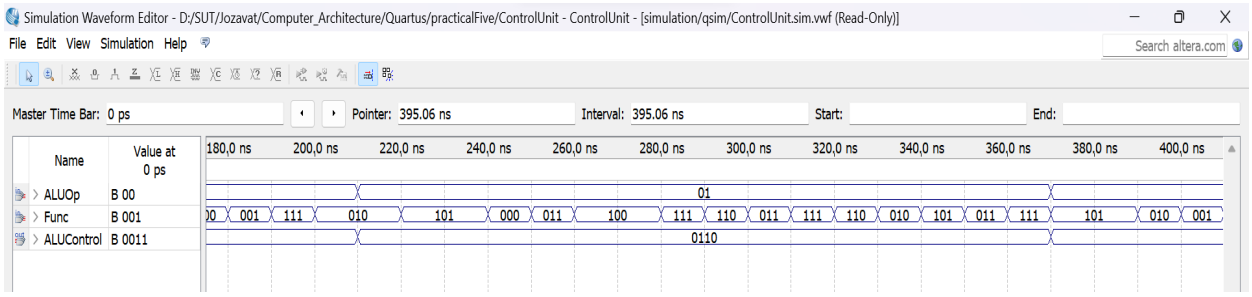
در جدول فوق، دستورات نوع Jump را در نظر نگرفته‌ایم زیرا دستورات ALU نیاز به عملیات ALU ندارند. شماتیک طراحی مدار فوق بصورت زیر است:



برای صحت‌سنجی مدار طراحی شده، آن را آزمایش کرده و به ازای مقادیر مختلف $ALUOp$ و $Func$ خروجی مدار را بررسی می کنیم.

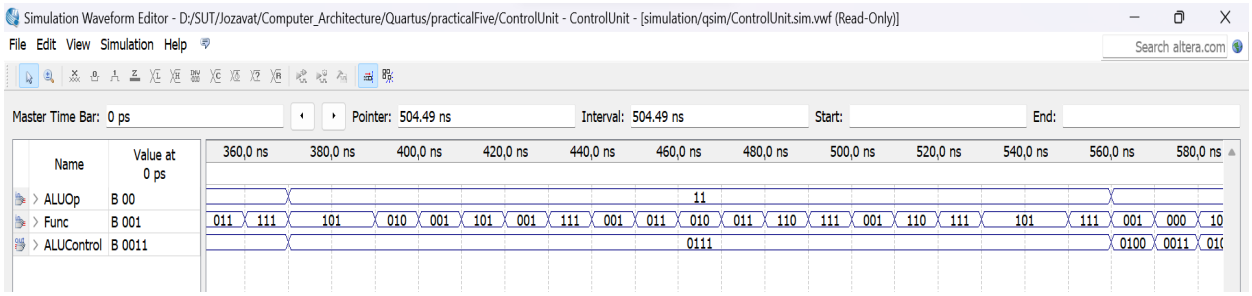


به ازای $ALUOp = 00$ مقدار ALUControl باید همواره 0011 باشد زیرا در دستور SB و SB، عملیات جمع در ALU انجام می‌شود.

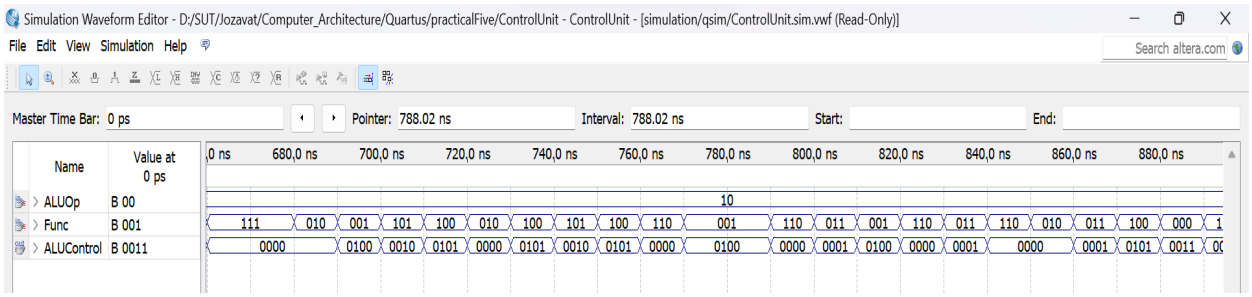


به ازای $ALUOp = 01$ مقدار ALUControl باید همواره 0110 باشد زیرا در دستور BEQ،

عملیات Equal در ALU انجام می‌شود.



به ازای $ALUOp = 11$، مقدار ALUControl باید همواره 0111 باشد زیرا در دستور BNQ، عملیات Not-Equal در ALU انجام می‌شود.



به ازای $ALUOp = 10$ دستورات R-format اجرا می‌شوند و بر اساس Func آن‌ها باید ALUControl را تعیین کنیم. طبق جدول فوق به ازای مقادیر مشخص شده، خروجی مورد نظر تولید شده است. همچنین به ازای مقادیری که هرگز اتفاق نمی‌افتند، مقدار ALUControl برابر 0000 است؛ برای مثال $Func = 111$ خروجی برابر 0000 شده است.

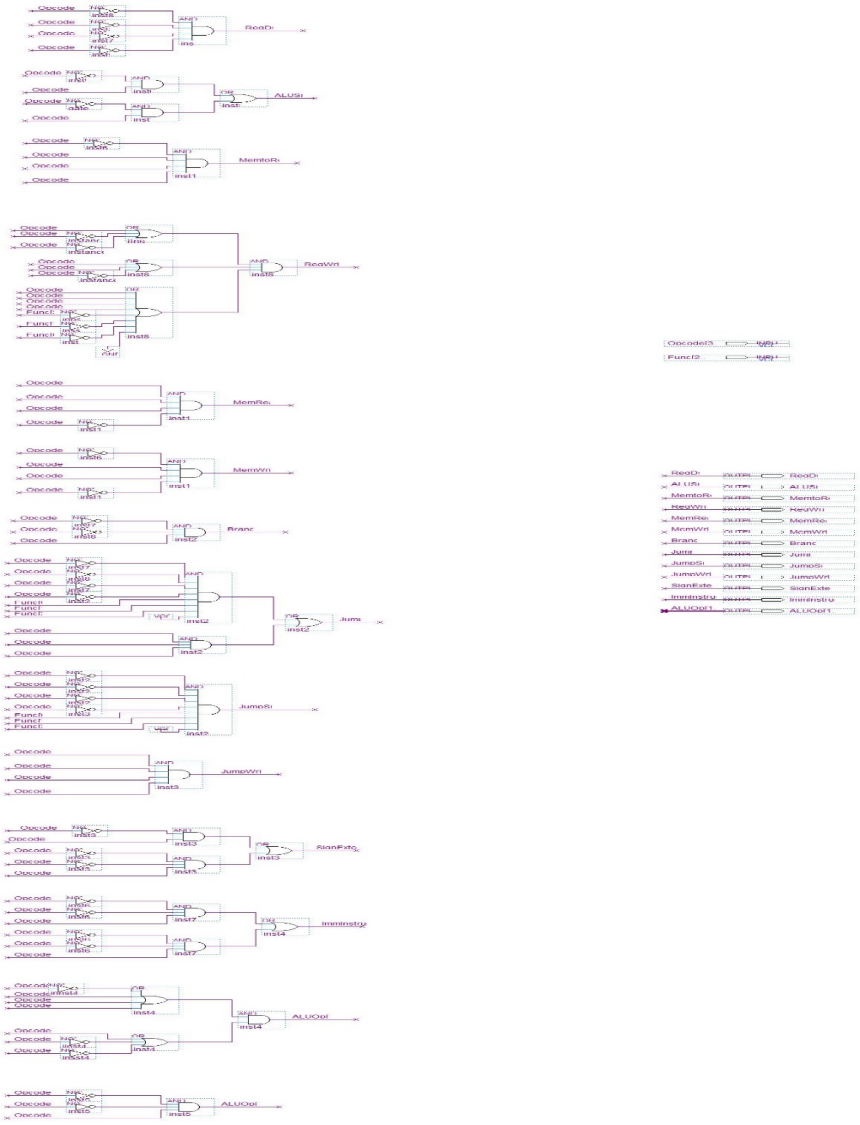
۲ Control Unit

برای طراحی واحد کنترل، باید ابتدا توابع منطقی سیگنال‌های کنترلی آن را به ازای ورودی $Opcode$ با کمک جدول کارنو بدست آورده و سپس شماتیک نهایی مدار را طراحی کرد.

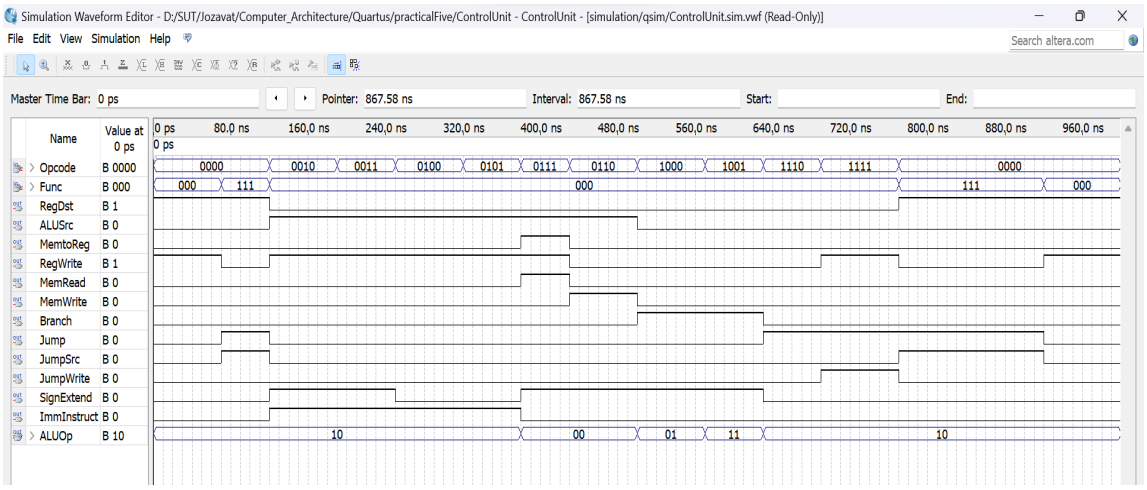
		R-format	ADDi	SUBi	ANDi	ORi	LB	SB	BEQ	BNQ	J	JAL	JR
	Op3	0	0	0	0	0	0	0	1	1	1	1	0
	Op2	0	0	0	1	1	1	1	0	0	1	1	0
	Op1	0	1	1	0	0	1	1	0	0	1	1	0
	Op0	0	0	1	0	1	1	0	0	1	0	1	0
Output	RegDst	1	0	0	0	0	0	X	X	X	X	X	X
	ALUSrc	0	1	1	1	1	1	1	0	0	X	X	X
	MemtoReg	0	0	0	0	0	1	X	X	X	X	X	X
	RegWrite	1	1	1	1	1	1	0	0	0	0	1	0
	MemRead	0	0	0	0	0	1	0	0	0	0	0	0
	MemWrite	0	0	0	0	0	0	1	0	0	0	0	0
	Branch	0	0	0	0	0	0	0	1	1	0	0	0
	Jump	0	0	0	0	0	0	0	0	0	1	1	1
	JumpSrc	X	X	X	X	X	X	X	X	X	0	0	1
	JumpWrite	0	0	0	0	0	0	0	0	0	0	1	0
	SignExtend	0	1	1	0	0	1	1	1	1	X	X	X
	ImmInstruct	0	1	1	1	1	0	0	0	0	0	0	0
	ALUOp1	1	1	1	1	1	0	0	0	1	X	X	X
	ALUOp0	0	0	0	0	0	0	0	1	1	X	X	X

$$RegDst = AND(Op3, Op2, Op1, Op0)$$
$$ALUSrc = OR(AND(Op3, Op2), AND(Op3, Op1))$$
$$MemToReg = AND(Op3, Op2, Op1, Op0)$$
$$RegWrite =$$
$$AND(OR(Op3, Op2, Op1), OR(Op2, Op1, Op0), OR(Op3, Op2, Op1, Op0, Func2, Func1, Func$$
$$MemRead = AND(Op3, Op2, Op1, Op0)$$
$$MemWrite = AND(Op3, Op2, Op1, Op0)$$
$$Branch = AND(Op3, Op2, Op1)$$
$$Jump =$$
$$OR(AND(Op3, Op2, Op1), AND(Op3, Op2, Op1, Op0, Func2, Func1, Func0))$$
$$JumpSrc = AND(Op3, Op2, Op1, Op0, Func2, Func1, Func0)$$
$$JumpWrite = AND(Op3, Op2, Op1, Op0)$$
$$SignExtend = OR(AND(Op3, Op1), AND(Op3, Op2, Op1))$$
$$ImmInstruct = OR(AND(Op3, Op2, Op1), AND(Op3, Op2, Op1))$$
$$ALUOp1 = AND(OR(Op3, Op2, Op1, Op0), OR(Op3, Op2, Op1))$$
$$ALUOp0 = AND(Op3, Op2, Op1)$$

طراحی مدار برای سیگنال‌های کنترلی واحد کنترل، به ازای توابع منطقی بدست آمده، بصورت زیر است:



برای آزمایش درستی مدار طراحی شده، خروجی تولید شده توسط آن را به ازای مقادیر مختلف $Opcode$ها و $Func$ ها با جدول بدست آورده در قسمت قبلی مقایسه می‌کنیم.



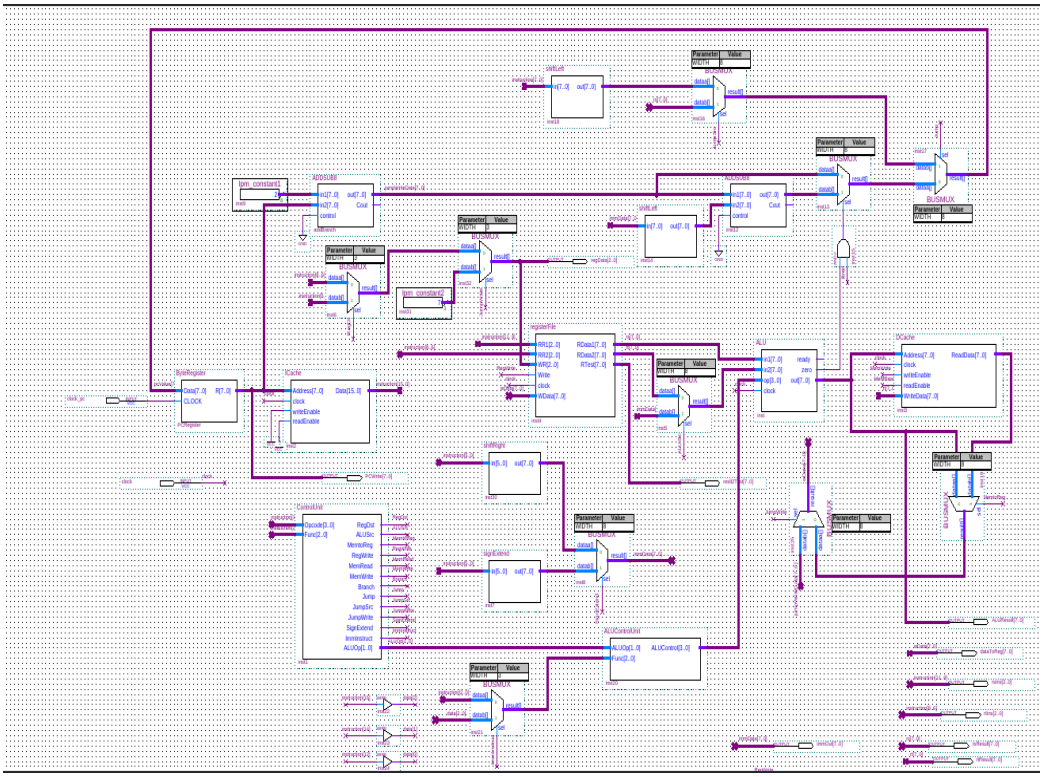
همانطور که مشاهده می کنید، به ازای تمام دستورات که $Opcode$ آنها در جدول فوق قرار دارد، خروجی مطلوب توسط مدار طراحی شده تولید شده است؛ برای مثال به ازای $Opcode = 0010$ مقادیر خروجی سیگنال‌های کنترلی بصورت زیر است:

Name	Value at 120.0 ns
> Opcode	B 0010
> Func	B 000
RegDst	B 0
ALUSrc	B 1
MemtoReg	B 0
RegWrite	B 1
MemRead	B 0
MemWrite	B 0
Branch	B 0
Jump	B 0
JumpSrc	B 0
JumpWrite	B 0
SignExtend	B 1
ImmInstruct	B 1
> ALUOp	B 10

طبق جدول بدست آمده در اول این بخش، خروجی سیگنال‌ها با سیگنال‌های درون جدول تطابق دارند.

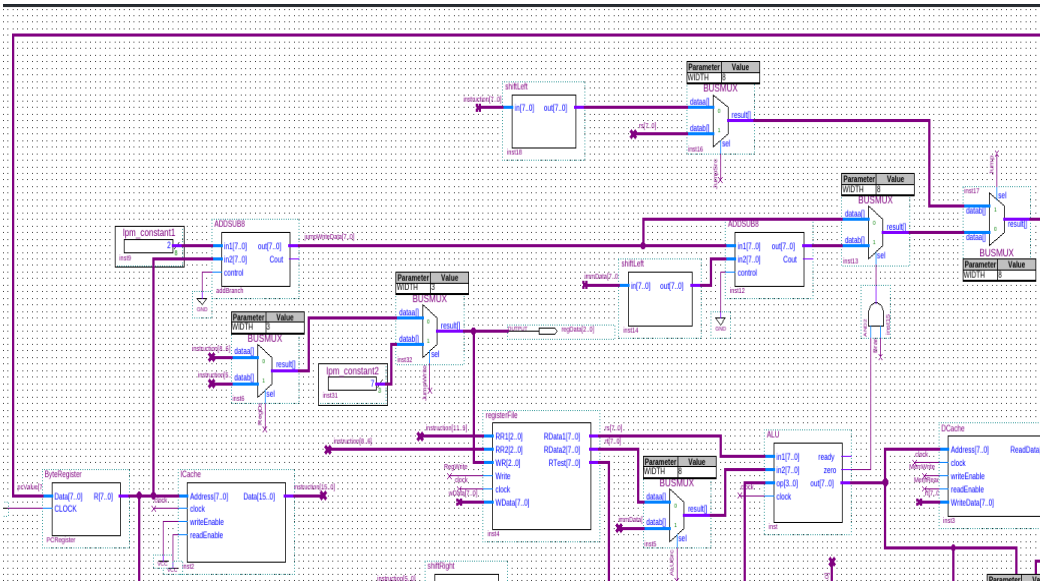
Datapath

شکل کلی datapath به صورت زیر می باشد که شامل ۴ بلوک instruction Memory, Reg-ister File, Data Memory, ALU و ثبات PC می باشد که ورودی و خروجی آنها با توجه به سیگنال های کنترلی مشخص شده است.



۱ Instruction Memory, PC register

ورودی ثبات PC توسط ۳ مالتی پلکسر تعیین می شود که با توجه به دستور فعلی و سیگنال های کنترلی Branch, JumpSrc, Jump مقدار PC تعیین می شود که می توان مقدار یک ثبات یا داده مستقیم درون آن ذخیره شود یا PC + 2 و یا با توجه به وضعیت Branch مقدار آن تعیین شود، خروجی ثبات PC به عنوان ورودی address به بلاک Instruction Memory داده می شود و دستور بعدی پردازش می شود. در شکل زیر این جزئیات آورده شده است.

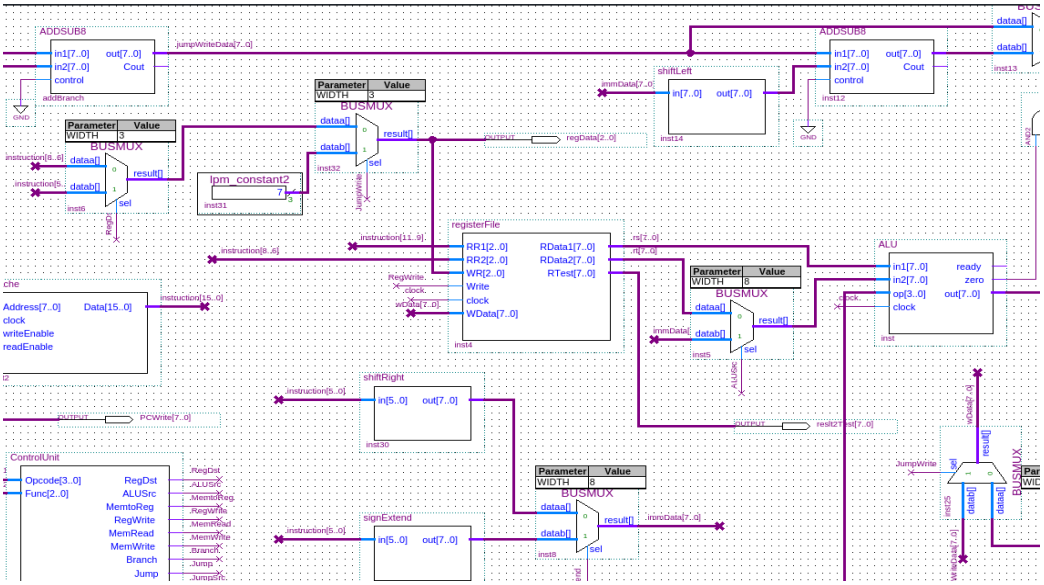


Register File, ALU ۲

ورودی های ثبات هایی که باید خوانده شوند توسط خروجی Instruction Memory تعیین می شوند و ثباتی که باید در آن نوشته شود توسط دو مالتی پلکسر و سیگنال های کنترلی -jump Write, RegDst تعیین می شود که می توان هر یک از مقادیر rt, rd و یا ۷ باشد که ۷ برای دستور JAL استفاده می شود که باید داخل ثبات هفتم نوشتن صورت بگیرد.

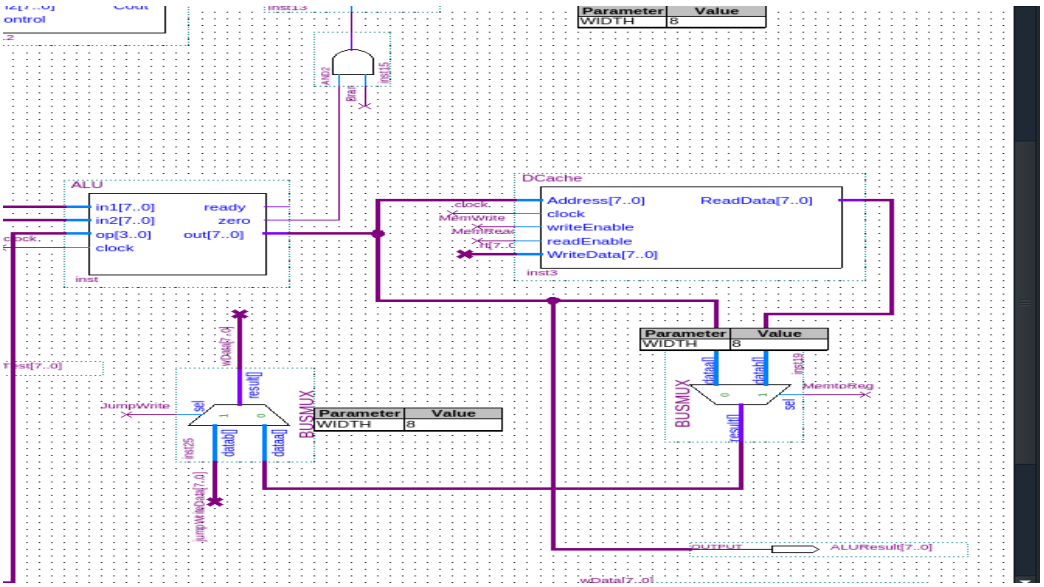
همچنین داده ای که باید ذخیره شود توسط دو مالتی پلکسر دیگر و سیگنال های کنترلی -jump Write, MemToReg تعیین می شود که می تواند مقدار خروجی Data Memory یا مقدار خروجی ALU و یا PC + 2 باشد، همچنین سیگنال RegWrite تعیین می کند که عمل نوشتن باید صورت بگیرد یا که مثدار ثباتی دست نخورد.

در بلاک ALU نیز ورودی های آن توسط دو مالتی پلکسر و سیگنال های کنترلی ALUSrc, SignExtend تعیین می شود که می تواند در واقع مقدار مستقیم ثبات دوم از خروجی register file باشد و یا مقدار Immediate که با توجه به دستورات این مقدار می توان sign extend باشد یا مقدار مستقیم آن باشد. در شکل زیر این جزئیات آورده شده است.



Data Memory ۳

ورودی addressاین قسمت توسط ALU تعیین می شود و همچین سیگنال های کنترلی ,MemWrite MemRead تعیین می کنند که چه عملی باید در این بلاک صورت بگیرد. در شکل زیر این جزئیات آورده شده است.



تست پردازنده

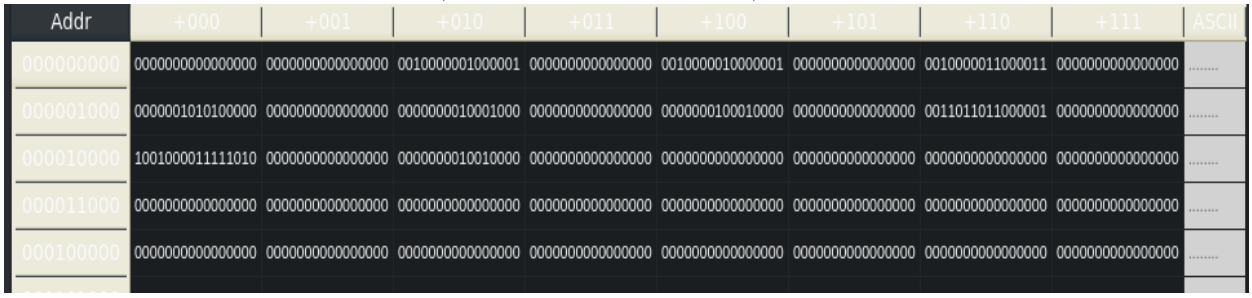
در این قسمت ۴ تست آورده شده است که دستورات مختلف در آن آورده شده است.

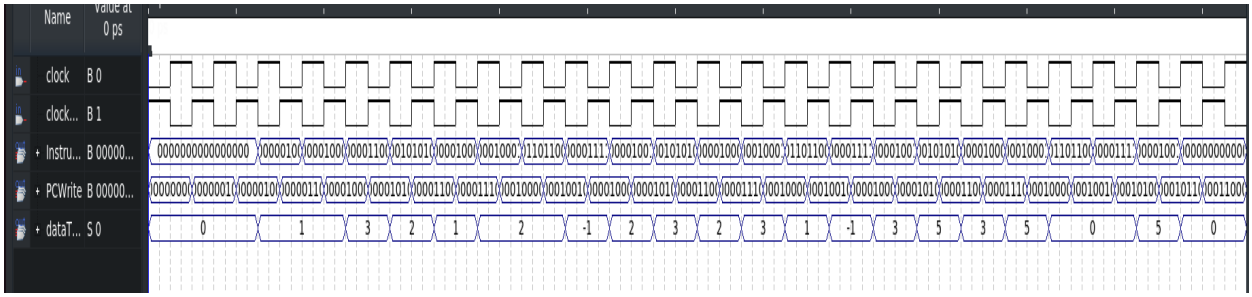
تست ۱

در این تست مقدار جمله پنجم دنباله فیبوناچی محاسبه شده است و داخل ثبات دوم ذخیره شده، دستورات انجام شده در این تست به صورت زیر است:

$ADDi\ R0, R1, 1$
$ADDi\ R0, R2, 1$
$ADDi\ R0, R3, 3$
$ADD\ R1, R2, R4$
$ADD\ R0, R2, R1$
$ADD\ R0, R4, R2$
$SUBi\ R0, R3, 1$
$BNQ\ R0, R3, -6$
$ADD\ R0, R2, R2$

که دستور آخر برای نمایش ثبات دوم و نشان دادن درستی پردازنده آورده شده است، در شکل های زیر این سری دستورات و نتیجه آن آورده شده است که همانطور که در شکل مشخص است در آخر مقدار داخل ثبات دوم ۵ است که برابر جمله پنجم دنباله فیبوناچی است.



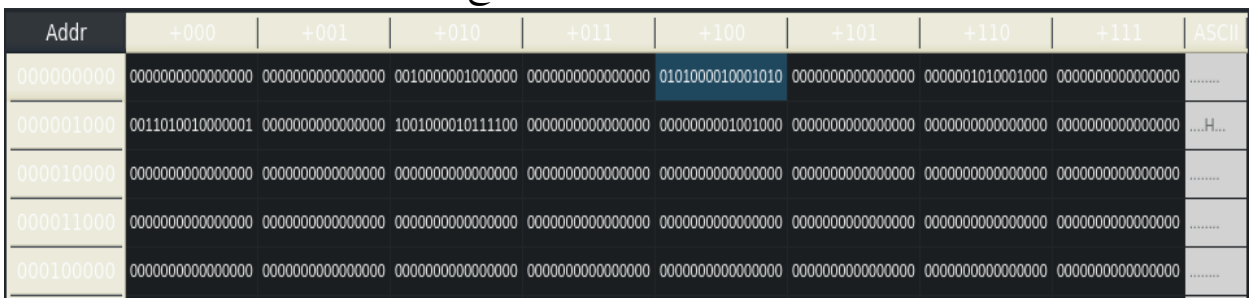


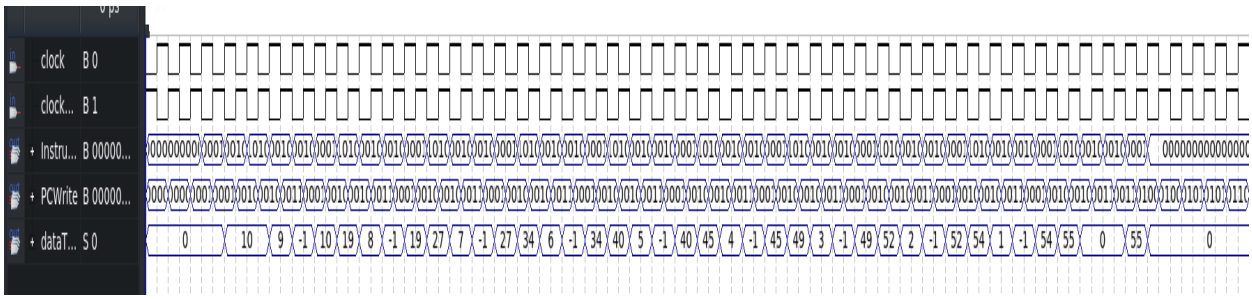
تست ۲

در این تست جمع اعداد ۱ تا ۱۰ داخل ثبات اول ذخیره شده است، دستورات انجام شده در این تست به صورت زیر است.

$ADDi\ R0, R1, 0$
$ORi\ R0, R2, 10$
$ADD\ R1, R2, R1$
$SUBi\ R2, R2, 1$
$BNQ\ R0, R2, -8$
$ADD\ R0, R1, R1$

که دستور آخر برای نمایش ثبات اول و نشان دادن درستی جواب پردازنده آورده شده است، در شکل های زیر این سری دستورات و نتیجه آن آورده شده است که همانطور که در شکل مشخص است در آخر مقدار ثبات اول برابر ۵۵ است که جمع اعداد ۱ تا ۱۰ می باشد.



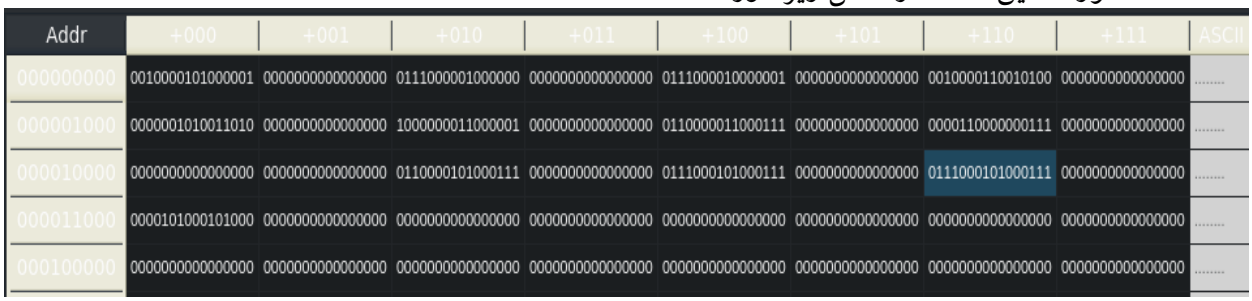


تست ۳

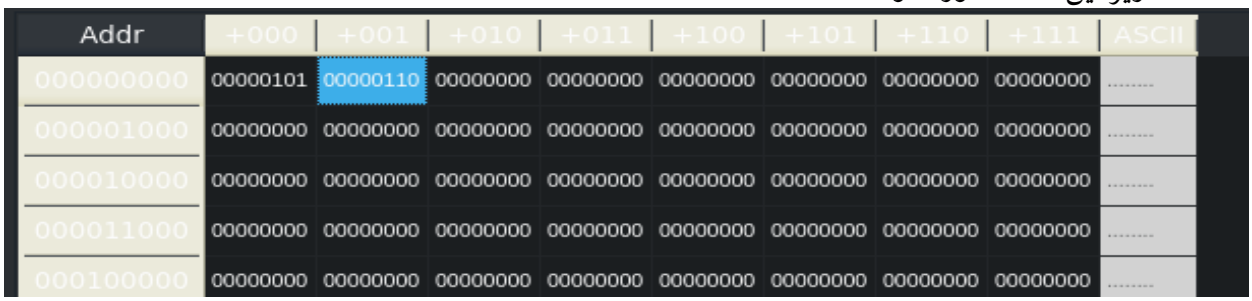
در این تست دو عددی که در خانه اول و دوم حافظه نوشته شده اند را با هم AND می کنیم و اگر جواب ۰ بود داخل خانه هفته حافظه عدد ۱ و در غیر این صورت حاصل AND این دو عدد را می نویسم، دستورات این تست به صورت زیر است:

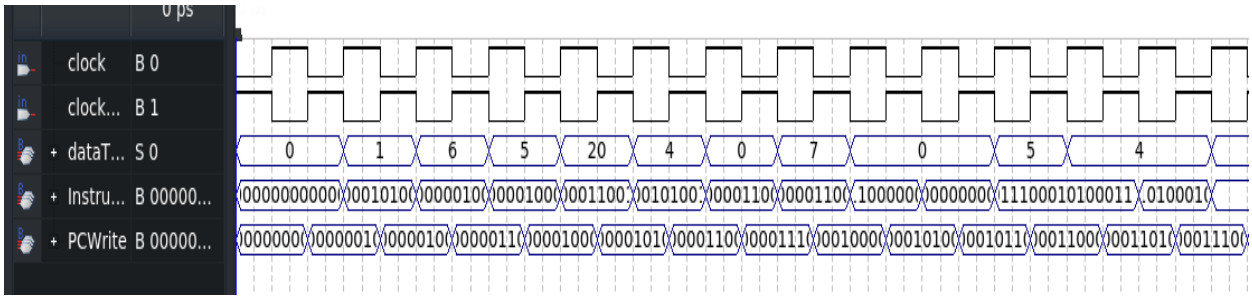
$ADDi\ R0, R5, 1$
$LB\ R0, R2, 0$
$LB\ R0, R1, 1$
$ADDi\ R0, R6, 20$
$AND\ R1, R2, R3$
$BEQ\ R0, R3, 2$
$SB\ R0, R3, 7$
$JR\ R6$
$SB\ R0, R5, 1$
$LB\ R0, R5, 1$
$ADD\ R0, R5, R5$

که دستور آخر برای نمایش ثبات پنجم و نشان دادن درستی پردازنده آورده شده است، سری دستورات این تست در شکل زیر آورده شده است:

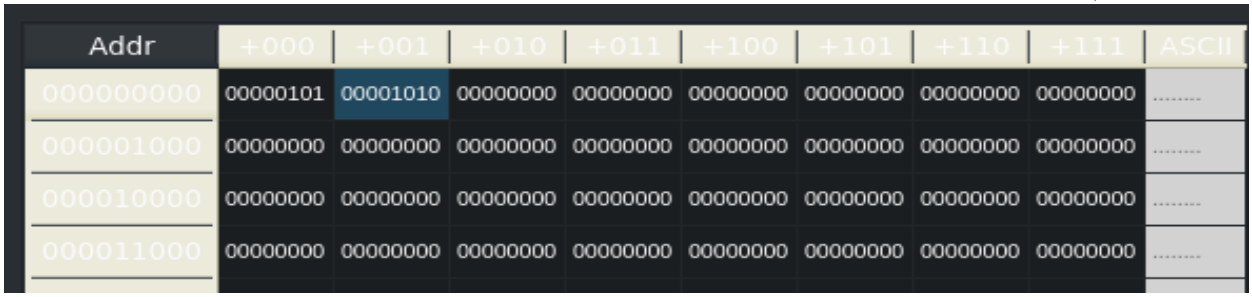


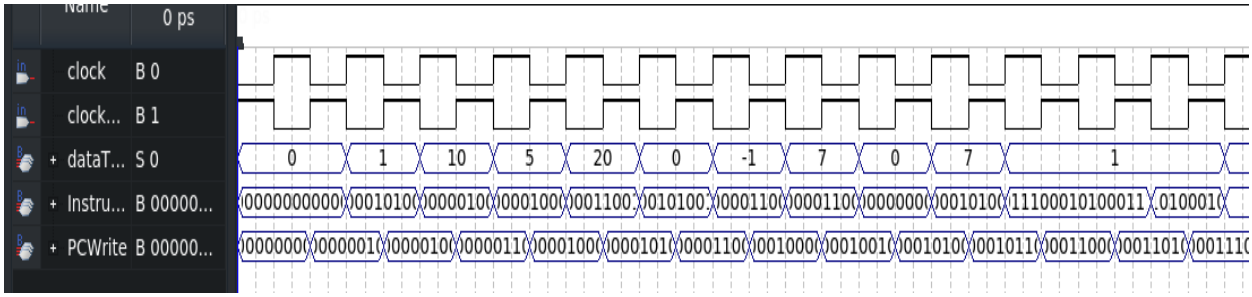
حال اگر داخل خانه های حافظه اعداد ۶ و ۵ باشد AND آنها ۴ می شود و در شکل های زیر این حالت آورده و تست شده است.





و اگر در خانه های اول و دوم حافظه ۵ و ۱۰ باشد AND آن ها ۰ می شود و باید داخل ثبات پنجم ۱ نوشته شود، در شکل های زیر این حالت آورده و تست شده است:





تست ۴

در این تست سری دستورات زیر انجام شده است:

$ADDi\ R0, R1, 15$
$ANDi\ R1, R2, 9$
$XOR\ R1, R2, R2$
$JAL\ 16$
$ADD\ R0, R0, R0$
$OR\ R0, R2, R2$
$OR\ R0, R2, R2$
$J\ 24$
$OR\ R0, R7, R7$
$OR\ R0, R7, R7$
$OR\ R0, R7, R7$
$ADD\ R0, R7, R7$
$OR\ R2, R7, R2$
$ADD\ R0, R2, R2$

که همانطور که در دستورات آمده است قبل از دستور JAL مقدار XOR عدد ۹ و ۱۵ داخل ثبات دوم ریخته می شود که برابر ۶ است و همچنین دستورات بین دستور Jump و JAL به دلیل پرش اجرا نمی شوند و محتویات ثبات دوم عوض نمی شود و همچنین پس از پرش دستور Jump سه دستور پس از آن اجرا نمی شود پس محتویات ثبات هفتم عوض نمی شود و همان $PC + 2$ می ماند که در دستور JAL این مقدار برابر $16 + 2 = 18$ است و در آخر ثبات دوم و هفتم با هم OR می شوند و نتیجه داخل ثبات دوم ریخته می شود که باید مقدار ۲۲ باشد.

در شکل زیر دستورات این تست و نتیجه آن آورده شده است که با توجه به خروجی های نشان داده شده در هر کلاک درستی پردازنده در این تست مشخص است.