به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

گزارش‌کار تمرین ششم

استاد: دکتر حسین اسدی

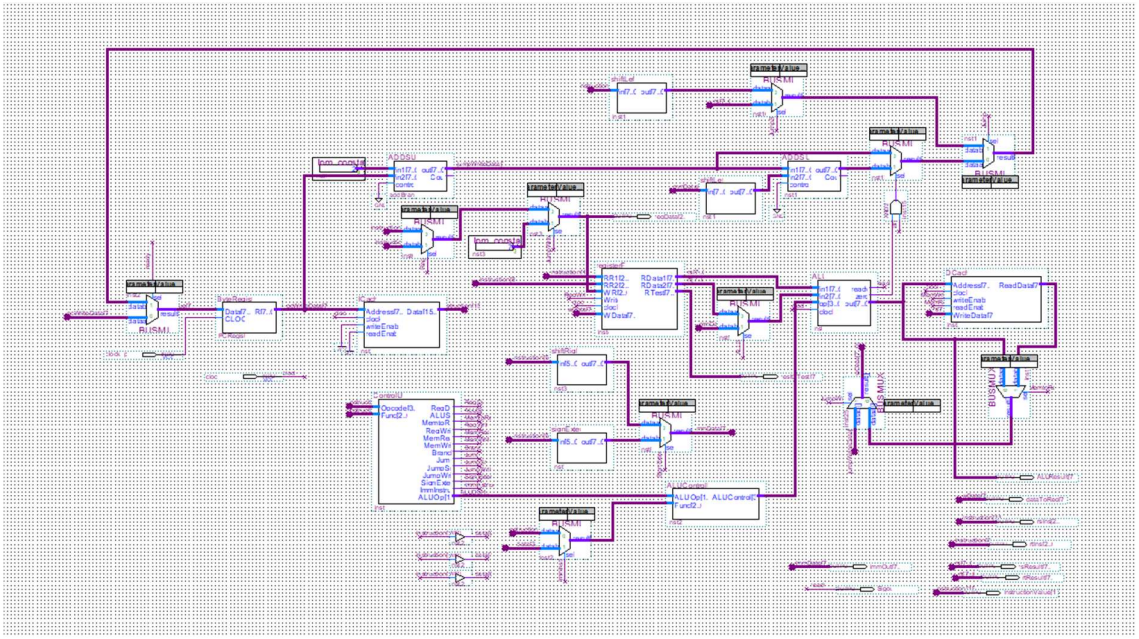
کیان قاسمی ۴۰۱۱۰۲۲۶۴

اشکان تاریوردی ۴۰۱۱۰۵۷۵۳

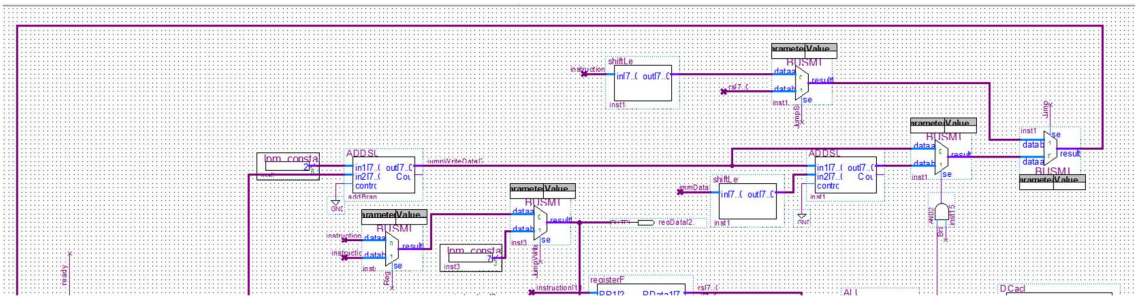
پیاده‌سازی پردازنده

برای انجام عملیات ضرب طبق الگوریتم Booth نیاز به ۸ چرخه ساعت (clock cycle) داریم که در تمارین قبلی پیاده‌سازی شده است. برای اضافه کردن این دستور به مجموعه دستورات پردازنده قبلی، نیاز به یک شمارنده (counter) داریم که از عدد ۱۵ شروع به شمارش می‌کند. به ۸ کلاک برای انجام ضرب و ۱ کلاک برای مقداردهی اولیه (initialize) نیاز داریم. هنگام رسیدن شمارنده به عدد ۶، مقدار سیگنال ready که به هنگام تمام شدن عملیات ضرب Booth ۱ می‌شد، ۱ می‌شود (سیگنال ready یک تابع منطقی از شمارنده است). همچنین نیاز به یک سیگنال کنترلی برای ضرب داریم که آن را isMul نام‌گذاری می‌کنیم. همچنین با توجه به دو بیت آخر خروجی هر مرحله از الگوریتم Booth، یکی از عملیات‌های جمع، تفریق یا No operation توسط مالتی‌پلکسر انتخاب می‌شود.

شکل کلی مسیر داده بصورت زیر است:



واحدهای مخصوص ضرب به روش الگوریتم Booth در شکل زیر نمایش داده شده‌اند:



تست پردازنده

ابتدا چهار تست قبلی که بر روی پردازنده‌ی قبلی انجام شده بود را بر روی این پردازنده جدید اجرا می‌کنیم.

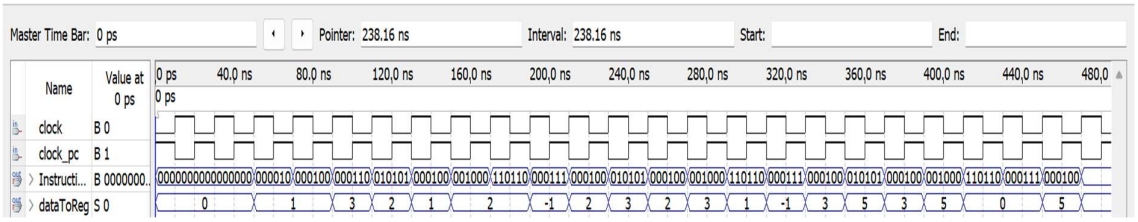
تست ۱

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.

Addr	+000	+001	+010	+011	+100	+101	+110	+111	ASCII
000000000	0000000000000000	0000000000000000	0010000001000001	0000000000000000	0010000010000001	0000000000000000	0010000011000011	0000000000000000	.....
000001000	0000001010100000	0000000000000000	0000000010001000	0000000000000000	0000000100010000	0000000000000000	0011011011000001	0000000000000000	.....
000010000	1001000011111010	0000000000000000	0000000010010000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	.....
000011000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	.....
000100000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	.....



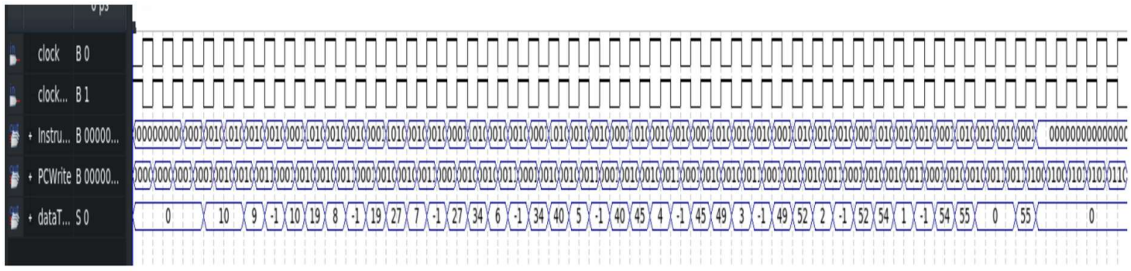
تست ۲

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 براي نمايش ثبات اول و نشان دادن درستي جواب پردازنده آورده شده است، در شكل هاي زير اين سري دستورات و نتيجه آن آورده شده است كه همانطور كه در شكل مشخص است در آخر مقدار ثبات اول برابر ۵۵ است كه جمع اعداد ۱ تا ۱۰ مي باشد.

Addr	+000	+001	+010	+011	+100	+101	+110	+111	ASCII
000000000	0000000000000000	0000000000000000	0010000001000000	0000000000000000	0101000010001010	0000000000000000	0000001010001000	0000000000000000	.....
000001000	0011010010000001	0000000000000000	1001000010111100	0000000000000000	0000000001001000	0000000000000000	0000000000000000	0000000000000000	...H...
000010000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	.....
000011000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	.....
000100000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	.....

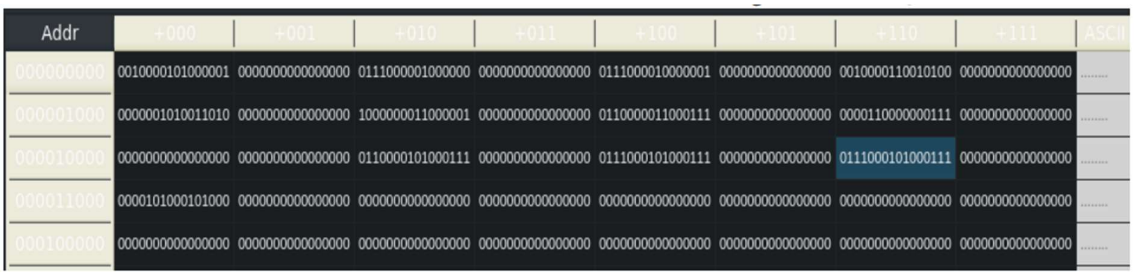


تست ۳

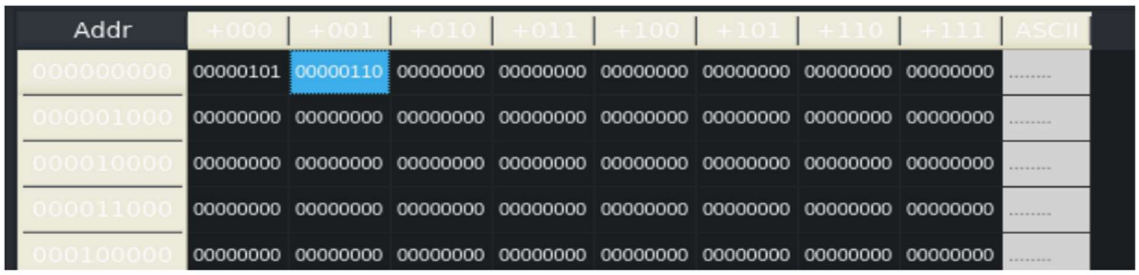
در این تست دو عددی که در خانه اول و دوم حافظه نوشته شده اند را با هم AND می کنیم و اگر جواب ۰ بود داخل خانه هفته حافظه عدد ۱ و در غیر این صورت حاصل AND این دو عدد را می نویسم، دستورات این تست به صورت زیر است:

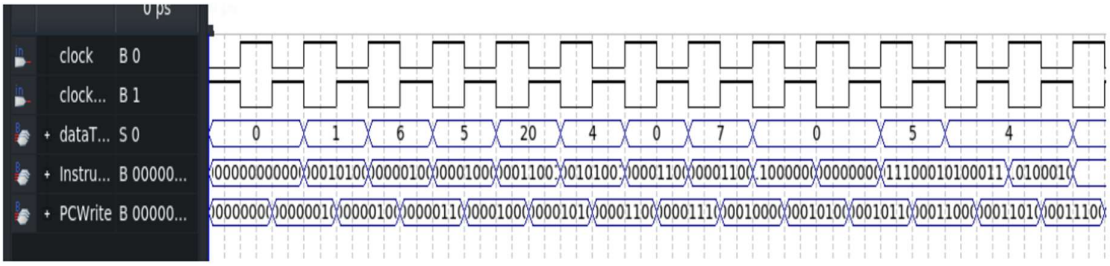
ADDi R0, R5, 1
LB R0, R2, 0
LB R0, R1, 1
ADDi R0, R6, 20
AND R1, R2, R3
BEQ R0, R3, 2
SB R0, R3, 7
JR R6
SB R0, R5, 1
LB R0, R5, 1
ADD R0, R5, R5

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:

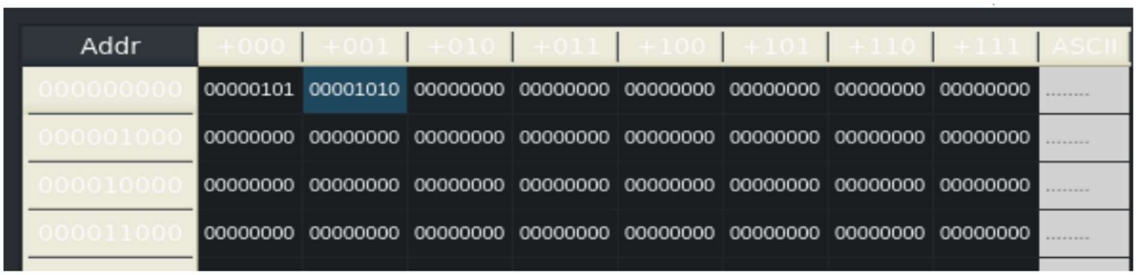


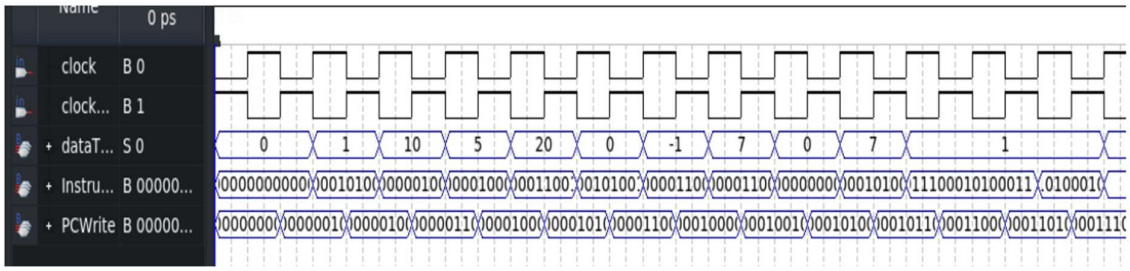
حال اگر داخل خانه های حافظه اعداد ۶ و ۵ باشد AND آنها ۴ می شود و در شکل های زیر این حالت آورده و تست شده است.





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:





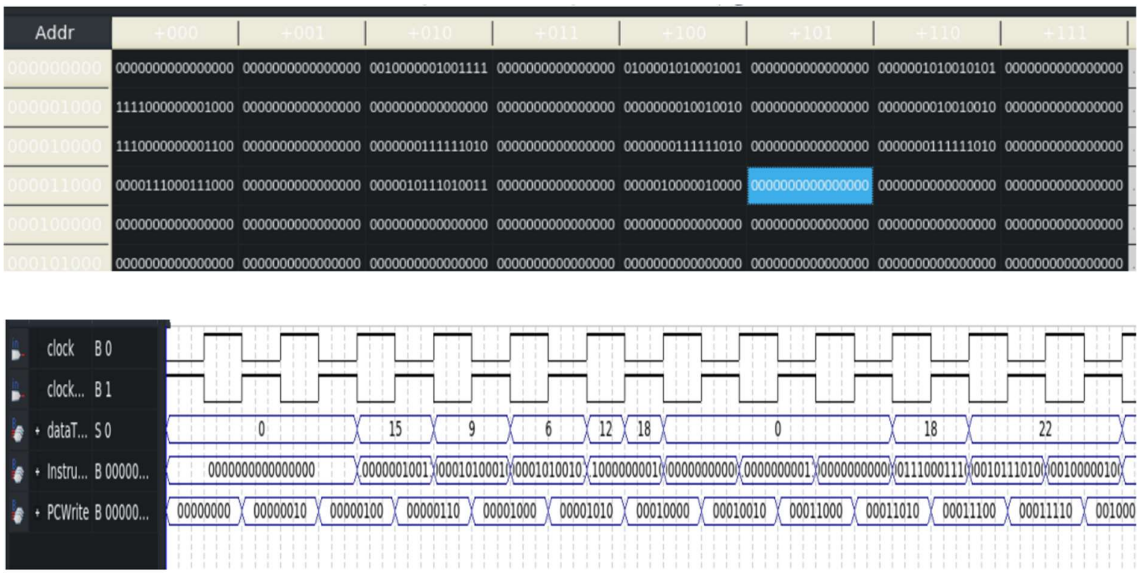
تست ۴

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 Jumpسه دستور پس از آن اجرا نمی شود پس محتویات ثبات هفتم عوض نمی شود و همان PC + 2می ماند که در

دستور JAL این مقدار برابر ۱۸ = ۲ + ۱۶ 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.

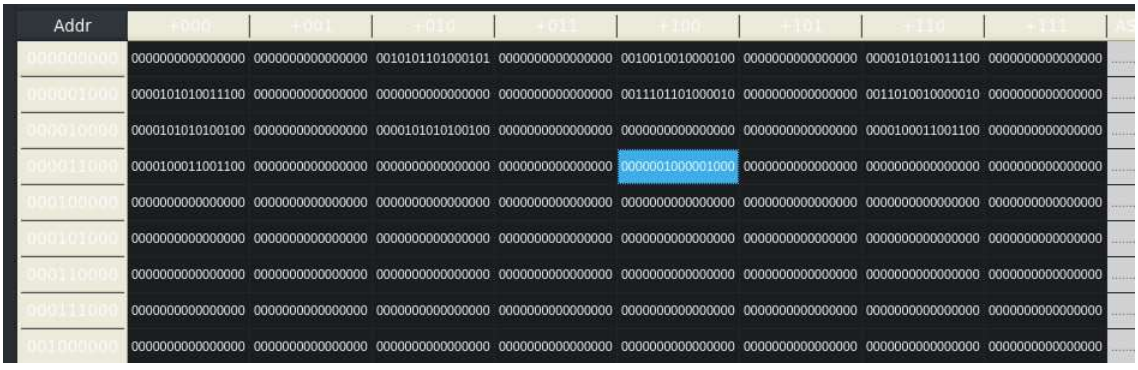


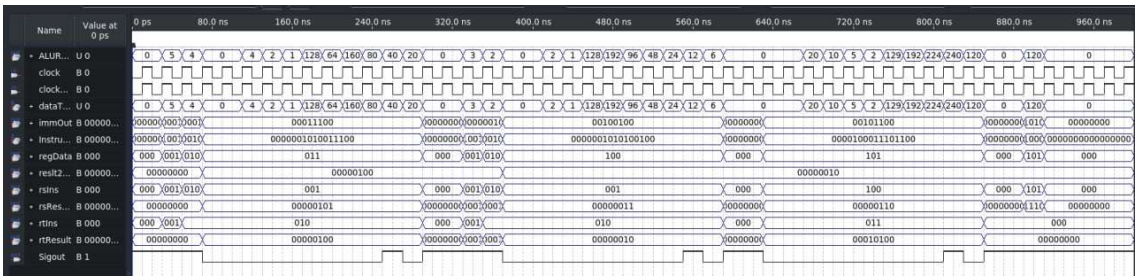
تست ۵

در این تست قرار است فاکتوریل عدد ۵ را محاسبه کنیم و نتیجه آن که برابر ۱۲۰ است را در ثبات ۱ ذخیره کنیم. بدین منظور دستورات زیر اجرا می‌شوند:

ADDi R0, R5, 5
ADDi R0, R2, 4
MUL R3, R5, R2
SUB R1, 2
SUB R2, 2
MUL R4, R5, R2
MUL R1, R4, R3
ADD R1, R0, R1

که دستور آخر به منظور نشان دادن مقدار ثبات R1 است.





همانطور که مشاهده می‌کنید، در ثبات ۱ مقدار ۱۲۰ که برابر ۵ فاکتوریل است ذخیره شده است.