**به نام خدا**

گزارشکار فاز چهارم پروژه درس معماری کامپیوتر

**پیاده­سازی Branch Prediction**

استاد

**دکتر حمید سربازی آزاد**

اعضای گروه

**محمدپارسا بشری 400104812**

**محسن قاسمی 400105166**

**امیرحسین رازلیقی 99102423**

بهار 1402

فهرست

[مقدمه و هدف فاز چهارم 2](#_Toc139477245)

[طرز کار Saturation Counter 2](#_Toc139477246)

[پیاده سازی Saturation Counter 2](#_Toc139477247)

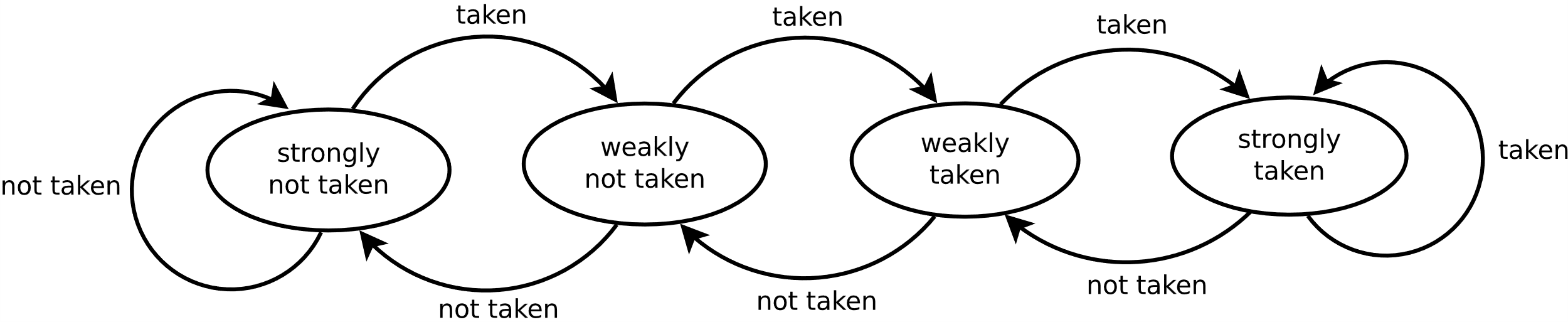
[تست عملکرد Branch Predictor 4](#_Toc139477248)

[منابع و مراجع 5](#_Toc139477249)

# مقدمه و هدف فاز چهارم

هدف کلی این پروژه، طراحی و پیاده­سازی یک پردازنده MIPS است. در فاز اول Datapath و Control Unit این پردازنده را به صورت Single Cycle طراحی و پیاده­سازی کردیم. در فاز دوم یک ماژول حافظه نهان (Cache) به پردازنده­مان اضافه کردیم. در فاز سوم پردازنده خود را از حالت Single Cycle به حالت Pipelined تغییر دادیم. در این فاز می­خواهیم با استفاده از branch prediction، مخاطرات کنترلی پایپلاین (control hazards) را برطرف کنیم. ایده اصلی این فاز پیش­بینی انجام یا عدم انجام پرش قبل از مشخص شدن واقعی تصمیم پرش است. در این فاز از saturation counter استفاده می­کنیم که توضیحات بیشتر در ادامه خواهد آمد.

# طرز کار Saturation Counter

استفاده از saturation counter به ما کمک می­کند که با توجه به پرش­های قبلی، بتوانیم انجام یا عدم انجام پرش را پیش­بینی کنیم. Saturation counter یک شمارنده دو بیتی است که طبق شکل زیر، با هر بار پرش مقدار خودش را آپدیت می­کند.

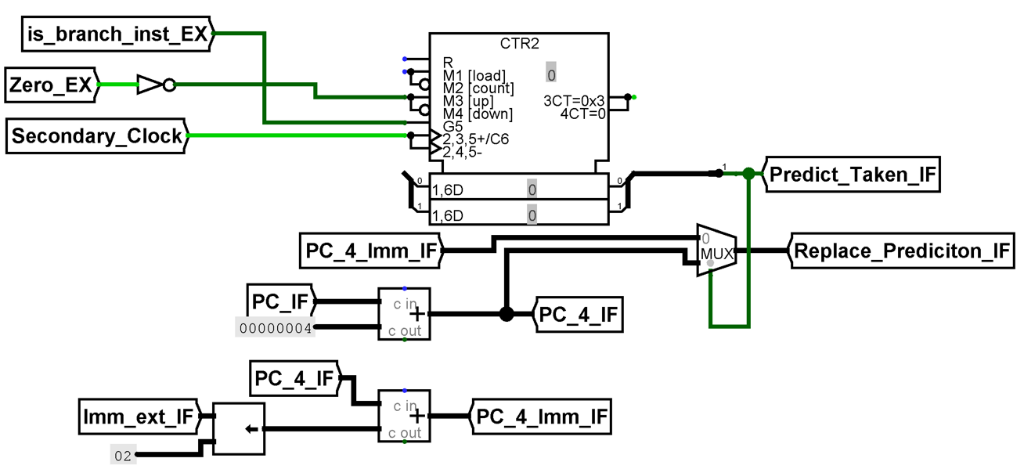
Predicted Taken

Predicted Not Taken

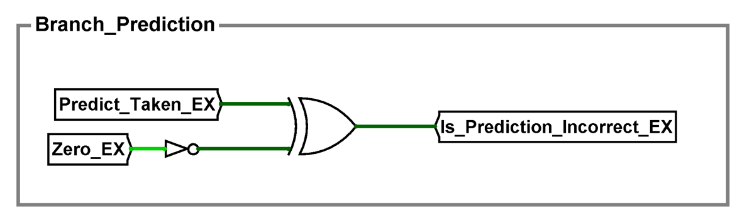
شکل 1- نحوه کار saturation counter

برای استفاده از این شمارنده باید به حالت فعلی آن توجه کنیم. اگر در یکی از حالات weakly taken یا strongly taken قرار داشتیم، پیش­بینی می­کنیم که پرش بعدی انجام خواهد شد؛ اما اگر در حالات weakly not taken و یا strongly not taken بودیم، پیش­بینی خواهیم کرد که پرش اتفاق نخواهد افتاد و خط بعدی برنامه اجرا خواهد شد (این پیش­بینی به راحتی با نگاه کردن به بیت سمت چپ شمارنده قابل انجام است). سپس وقتی به استیج EX رسیدیم و واقعا نتیجه پرش مشخص شد، با توجه به اینکه پیش­بینی­مان درست بوده یا خیر، saturation counter را آپدیت می­کنیم.

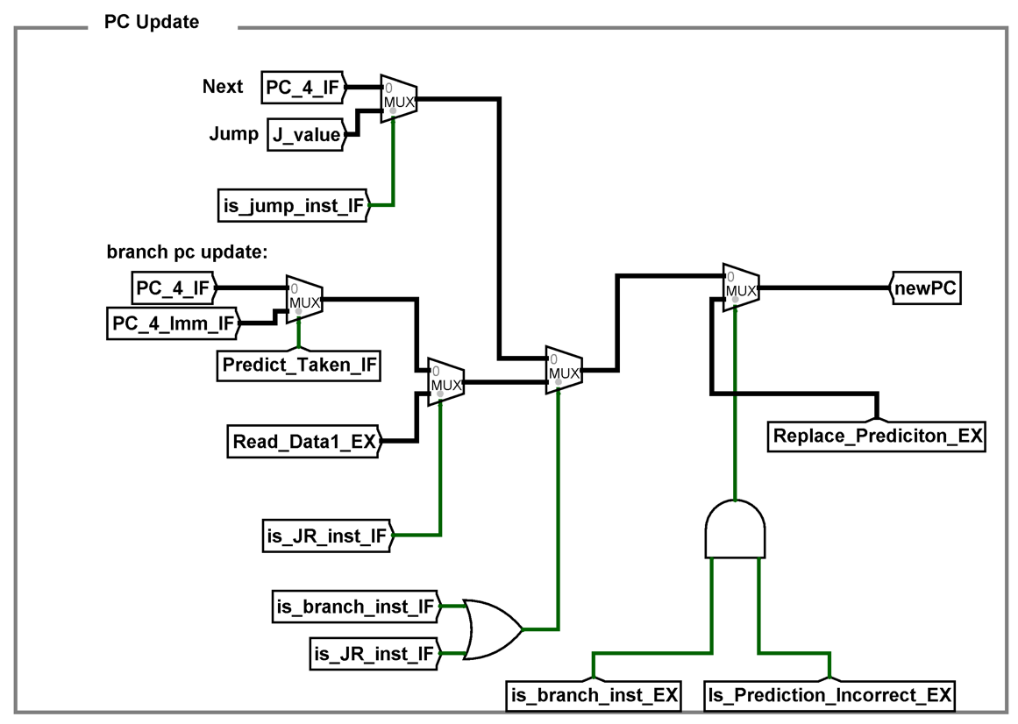
# پیاده سازی Saturation Counter

در اولین قدم باید خود شمارنده را پیاده­سازی کنیم. به جای شمارنده­ای که برای دستورات پرش شرطی (branch) در فاز قبل گذاشته بودیم، یک شمارنده جدید قرار می­دهیم و گزینه Action On Overflow آن را در حالت Stay at value می­گذاریم تا عملکرد مورد نظرمان در شکل 1 را داشته باشد. همچنین آدرس جایگزینی که تصمیم گرفته­ایم به آن پرش نکنیم را نیز تولید کرده و به داخل پایپلاین می­فرستیم.

شکل 2- پیاده­سازی saturation counter

حالا در استیج EX سیگنال Zero\_EX نشان می­دهد که آیا باید پرش انجام شود یا خیر. در این استیج تصمیمی که پیش­بینی کرده بودیم را با این سیگنال مقایسه می­کنیم و تشخیص می­دهیم که آیا پیش­بینی­مان درست بوده یا خیر.

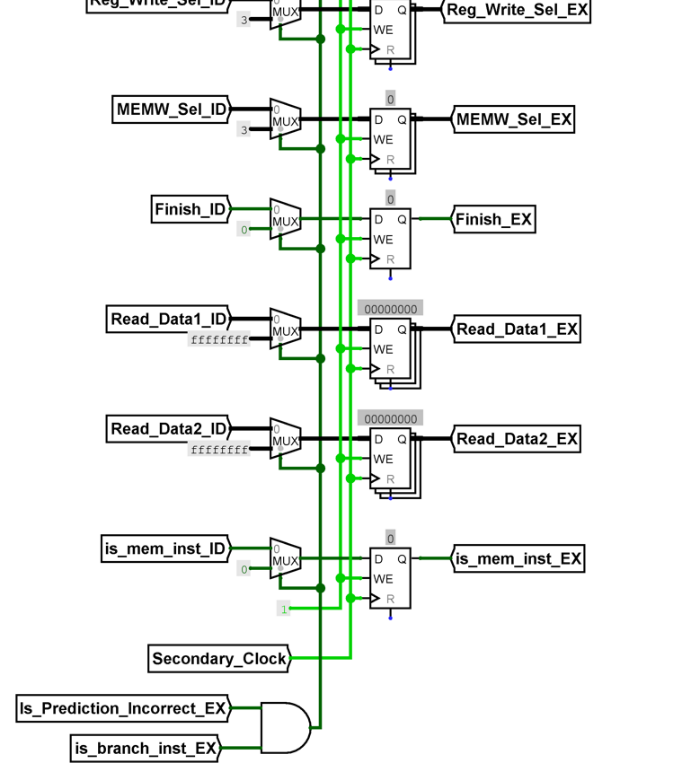
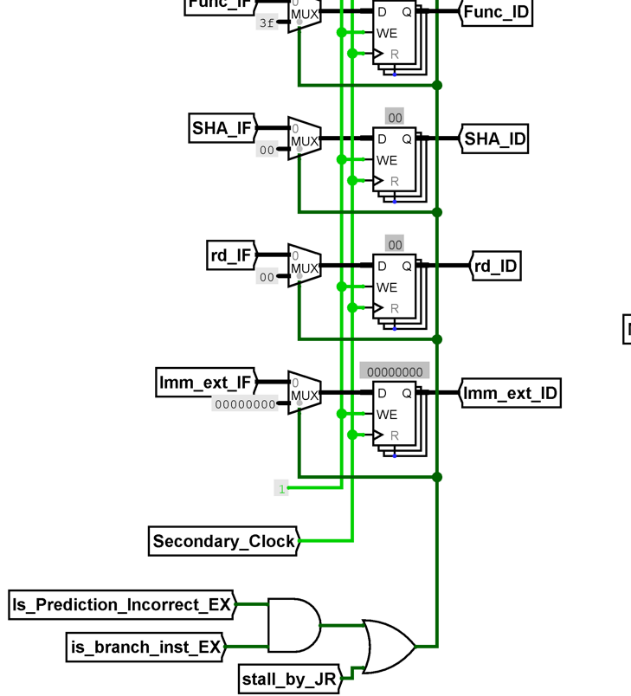
شکل 3- بررسی درست یا نادرست بودن پیش­بینی انجام شده

سپس با استفاده از این سیگنال، مقدار بعدی PC را مشخص می­کنیم.

شکل 4- منطق تولید مقدار جدید PC

همچنین در صورت پیش­بینی اشتباه باید پایپلاین را خالی کنیم. به تقلید از فرایند stall فاز قبل، در اینجا علاوه بر بافرهای IF-ID جلوی بافرهای ID-EX هم مالتی­پلکسر می­گذاریم. سیگنال­های کنترلی این مالتی­پلکسرها نیز به شکل زیر تغییر می­کنند.

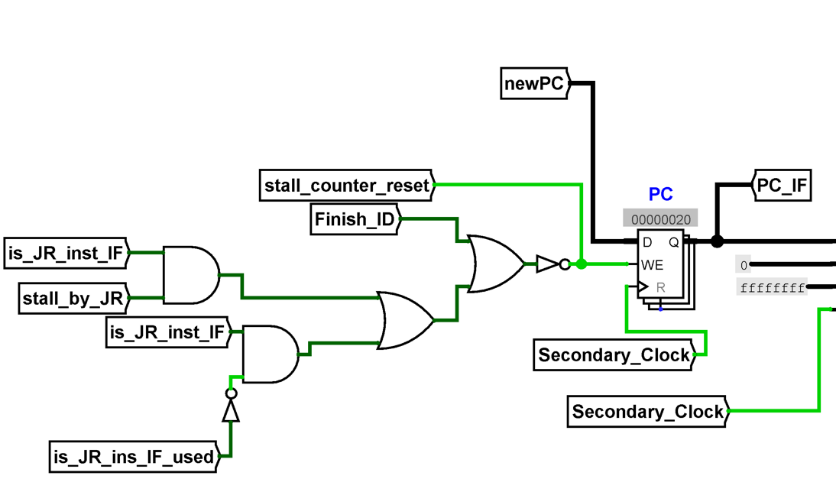
**ID**



**EX**

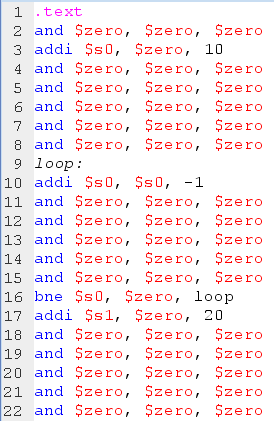
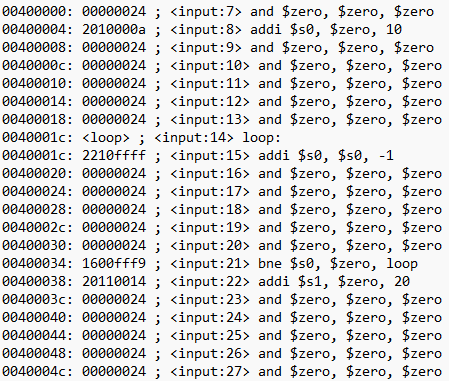
**IF**

شکل 5- کنترل بافرهای میانی

همچنین با توجه به اینکه در این فاز فقط دستور JR باعث ایجاد stall می­شود، مدار تولید WE برای PC را هم به شکل زیر تغییر می­دهیم:

شکل 6- منطق تولید WE برای PC

# تست عملکرد Branch Predictor

در این فاز از branch prediction استفاده کردیم تا مخاطرات کنترلی را برطرف کنیم. برای بررسی عملکرد درست این ابزار، برنامه زیر را روی پردازنده­مان اجرا می­کنیم.

شکل 8- کد ماشین برنامه شکل 7

شکل 7- برنامه جهت تست branch prediction

برنامه بالا 10 بار یک حلقه را تکرار می­کند. اگر این برنامه را بدون استفاده از branch prediction اجرا کنیم، زمان اجرای برنامه به شکل زیر محاسبه می­شود:

حالا اگر با استفاده از branch prediction برنامه را اجرا کنیم، در پیمایش اول و دوم و آخر پیش­بینی اشتباه رخ می­دهد که باعث می­شود زمان اجرای برنامه به شکل زیر محاسبه شود:

بنابراین میزان speedup حاصل از branch prediction به شکل زیر محاسبه می­شود:

پس مشاهده شد که branch prediction به رفع مخاطرات کنترلی و افزایش سرعت پردازنده کمک می­کند.

# نمره امتیازی: Forwarding

اینجا ادامه بدین...

# منابع و مراجع

* اسلایدهای درس
* <https://alanhogan.com/asu/assembler.php>
* <https://www.researchgate.net/figure/Two-bit-saturating-counter_fig3_221219835>
* <https://en.wikipedia.org/wiki/Branch_predictor>