به نام خدا

فاطمه شفیعی اردستانی

پریسا عسگرزاده

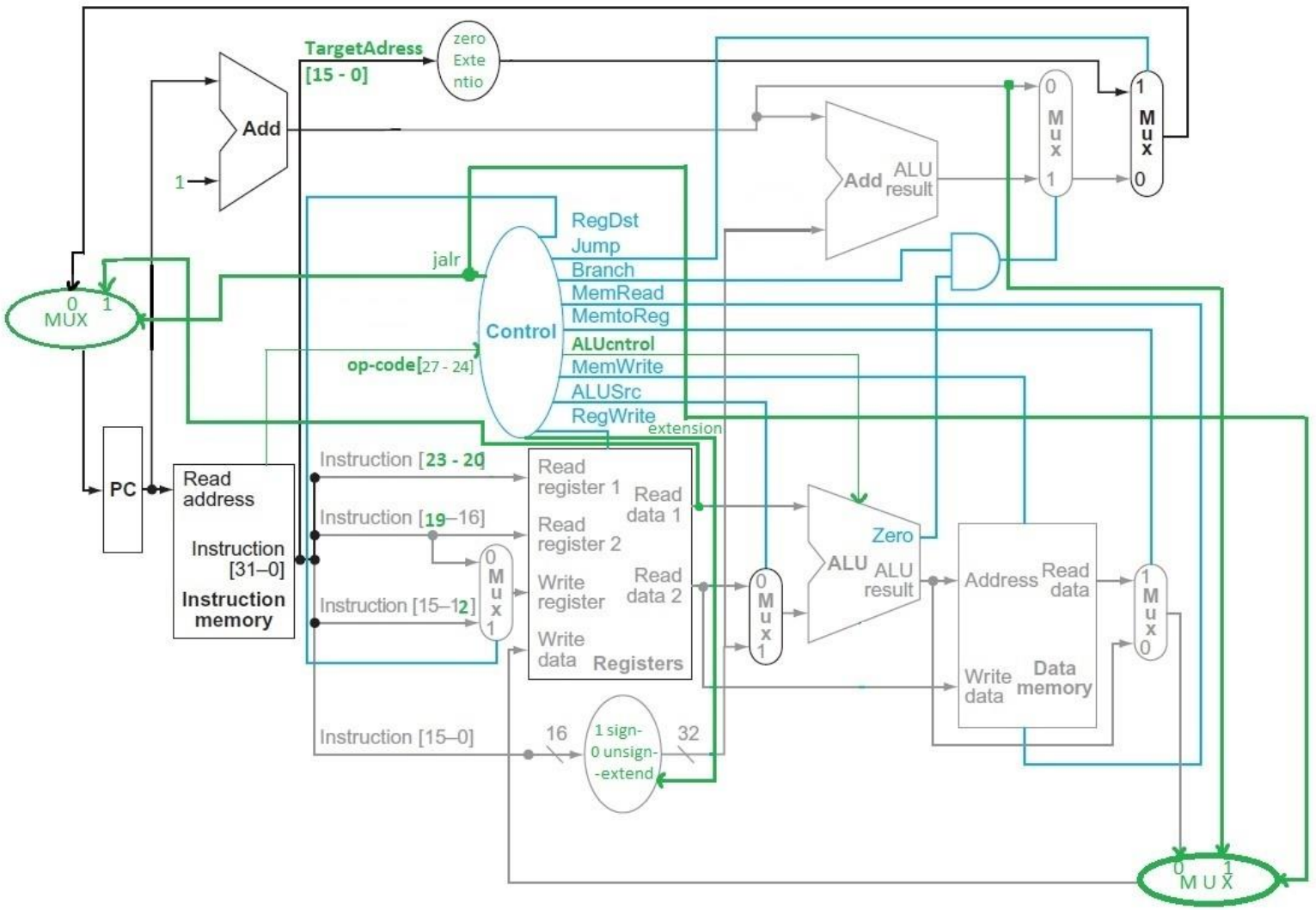
توضیحات اولیه پروژه

- این پروژه دارای دو بخش است که بخش اول آن، برنامه نوشته شده به زبان اسمبلی تعریف شده را اسمبل می‌کند و بخش دوم آن، برنامه اسمبل شده که یک سری اعداد ده-دهی (یا دو-دویی) است را به یک پردازنده شبیه سازی شده می‌دهد.

- بخش دوم پروژه، به صورت تک سیکلی پیاده سازی شده است. در پردازنده تک سیکلی، هردستور در یک سیکل پردازش می‌شود.

در اسلاید بعدی، مدار پردازنده تک‌سیکل را مشاهده می‌کنید.

طراحی مدار ماشین یک حلقه



توضیحات اضافه چگونگی کارکرد پردازنده تک حلقه

- دستورات اسمبل شده در ابتدا از حافظه آورده شده و سپس دی‌کد می‌شوند. بعد هربخشی از دستور۳۲بیتی، به مقصد موردنظر می‌رود. برای مثال آپ‌کد(۴ بیت) که به بخش کنترل می‌رود، سیگنال‌هایی تولید خواهد کرد که در بخش‌های دیگر پردازنده بتوانیم به درستی مقدار موردنظر را انتخاب کنیم.

- در اسلایدهای بعدی، چگونگی کاردکرد بخش کنترل را خواهید دید.

سیگنال های ارسالی از Control

- هرکدام از سیگنال‌های ارسال شده از کنترل، با یک اندیس شناخته می‌شوند. همچنین برای هر نوع از دستورات، در اسلایدهای آتی یک جدول خواهیم داشت که هر کنترل سیگنال و مقدارش مشخص شده‌اند.

	jalr	RegDst	Jump	Branch	MemRead	MemtoReg	AluCntol	MemWrite	AluSrc	Regwrite	extension
index	0	1	2	3	4	5	6	7	8	9	10

- برای سیگنال ALU، مقادیر به جای ۰ و ۱، از ۰ تا ۵ هستند به طوری که:

Add	0
Sub	1
Slt	2
Or	3
Nand	4
Shift	5

سیگنال‌های ارسالی برای دستورات R_Type

signals	jalr	RegDst	Jump	Branch	MemRead	MemtoReg	AluCntol	MemWrite	AluSrc	Regwrite	extension
Add	0	1	0	0	0	0	0	0	0	1	X
Sub	0	1	0	0	0	0	1	0	0	1	X
Slt	0	1	0	0	0	0	2	0	0	1	X
Or	0	1	0	0	0	0	3	0	0	1	X
Nand	0	1	0	0	0	0	4	0	0	1	X

I_Type سیگنال‌های ارسالی برای دستورات

signals	jalr	RegDst	Jump	Branch	MemRead	MemtoReg	AluCntol	MemWrite	AluSrc	Regwrite	extension
Addi	0	0	0	0	0	0	0	0	1	1	1
Ori	0	0	0	0	0	0	3	0	1	1	0
Slti	0	0	0	0	0	0	2	0	1	1	1
Lui	0	0	0	0	0	0	5	0	1	1	X
Lw	0	0	0	0	1	1	0	0	1	1	1
Sw	0	X	0	0	0	0	0	1	1	0	1
Beq	0	X	0	1	0	0	1	0	0	0	1
Jalr	1	0	0	0	0	0	X	0	0	1	X

سیگنال‌های ارسالی برای دستورات R_Type

signals	jalr	RegDst	Jump	Branch	MemRead	MemtoReg	AluCntol	MemWrite	AluSrc	Regwrite	extension
j	0	X	1	0	0	0	X	0	X	0	X
halt	0	X	0	0	0	0	X	0	X	0	X

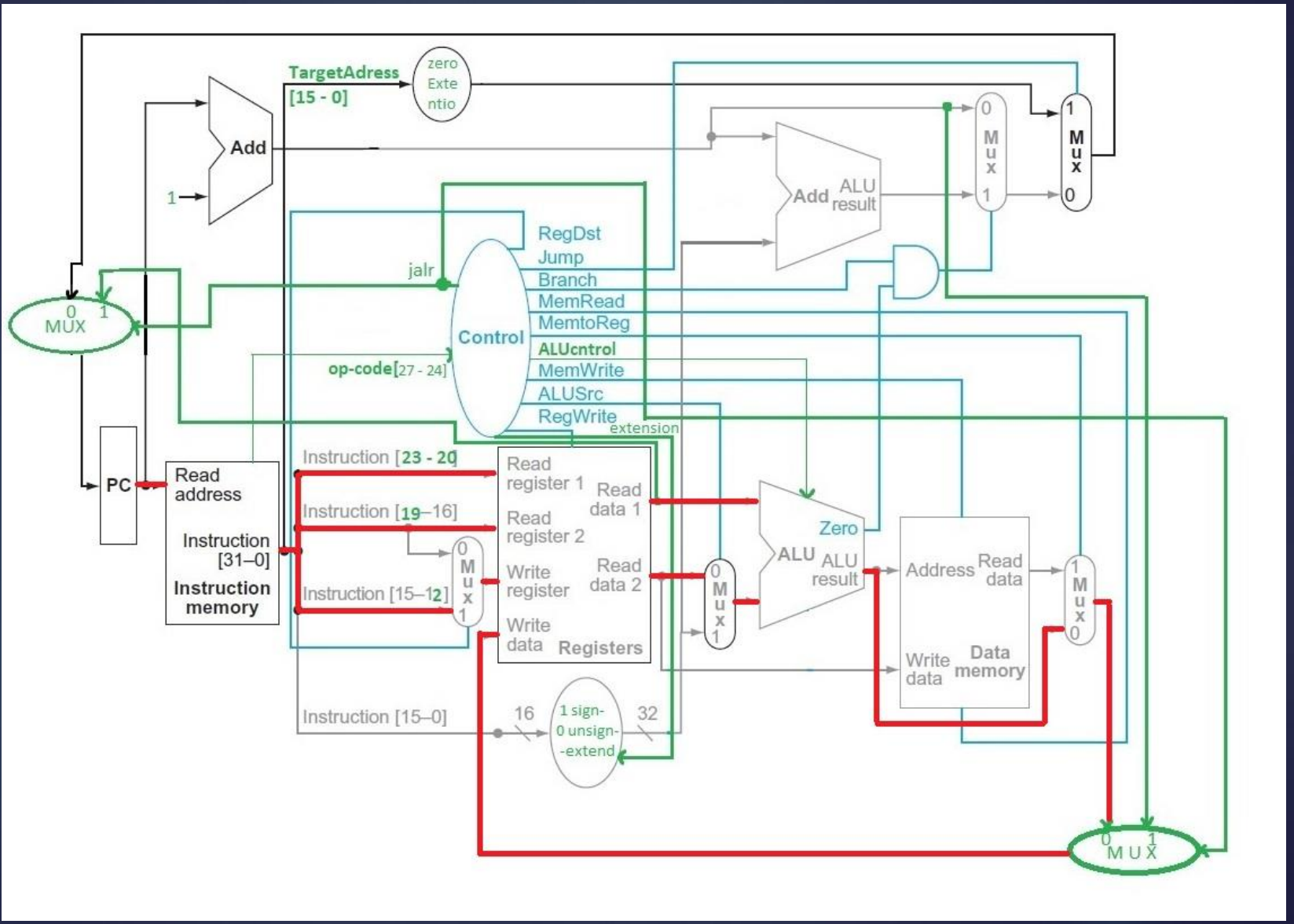
- در بخش instruction memory، رجیسترهای read خوانده شده و از بین دو رجیستر rd, rt رجیستر مقصد انتخاب می‌شود. این انتخاب و همچنین نوشتن در رجیستر انتخاب شده، توسط سیگنالی از کنترل مشخص می‌شود.

- پس از آن در ALU، عملیات موردنظر (که باز توسط قسمت کنترل مشخص می‌شود) انجام شده و به مقاصد مورد نظر می‌رود.
- در همین حین، آدرس دستورالعمل بعدی انتخاب می‌شود. این آدرس از بین ۴ انتخاب مختلف گذشته و مقدارنهایی آن مشخص می‌شود.

- در نهایت در صورتی که لازم باشد، مقداری را از واحد مموری گرفته و از بین مقدار گرفته شده از مموری و حاصل ALU (توسط سیگنالی از کنترل) یکی انتخاب شده و در صورت لزوم (الزام توسط سیگنالی از کنترل مشخص خواهد شد.) در رجیستر مقصد نوشته می‌شود.

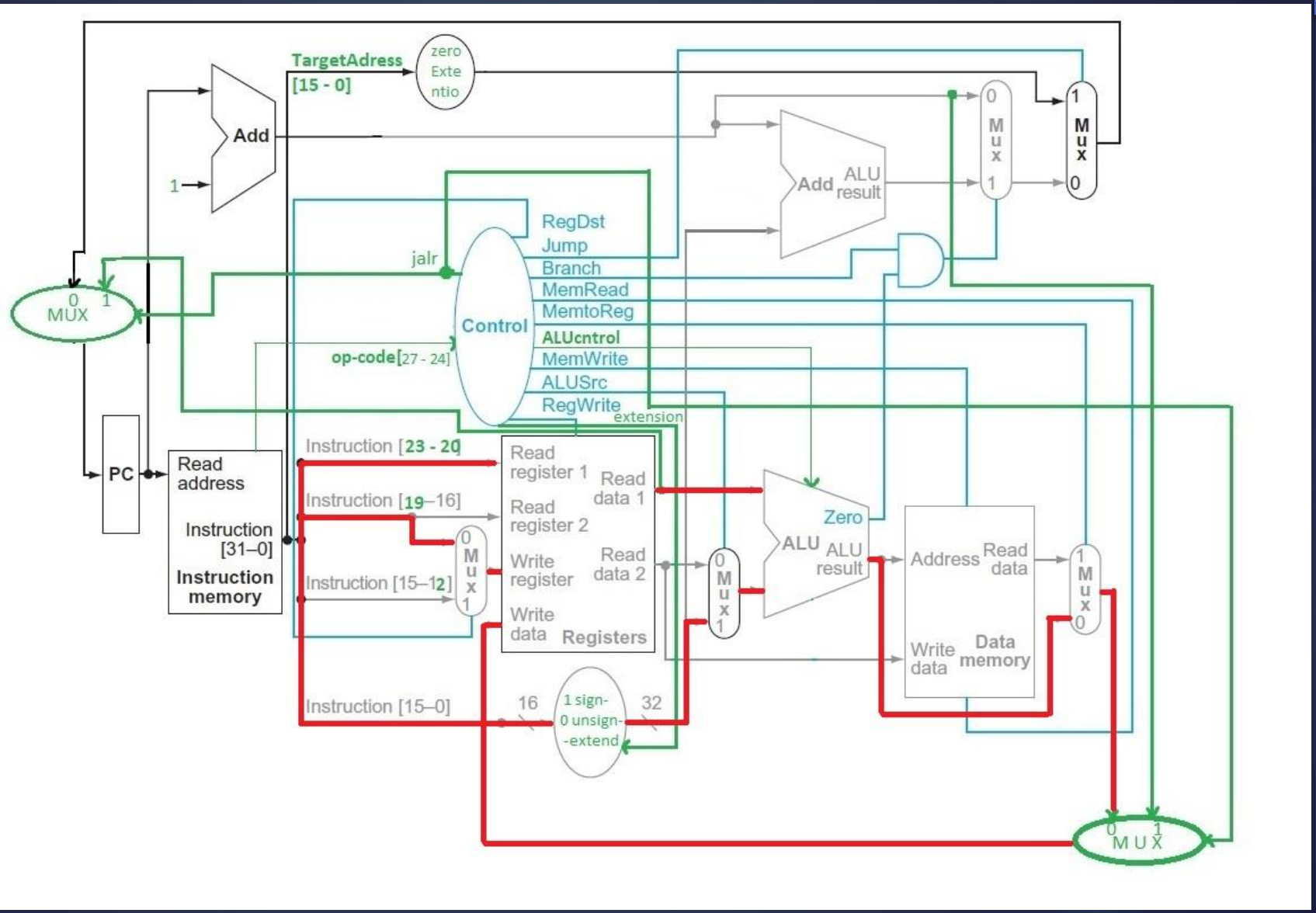
در این قسمت، مسیر یکی از دستورات R_Type با رنگ قرمز نشان داده شده است.

در این حالت، پی سی تغییر خاصی نمیکند و کلا +۱ می‌شود. همه سیگنال‌های کنترل که استفاده می‌شوند نیز علامت زده نشده است. با سیگنال RegWrite مقدار موردنظر (که توسط ALU بدست آمده است) در رجیستر مقصد نوشته می‌شود.



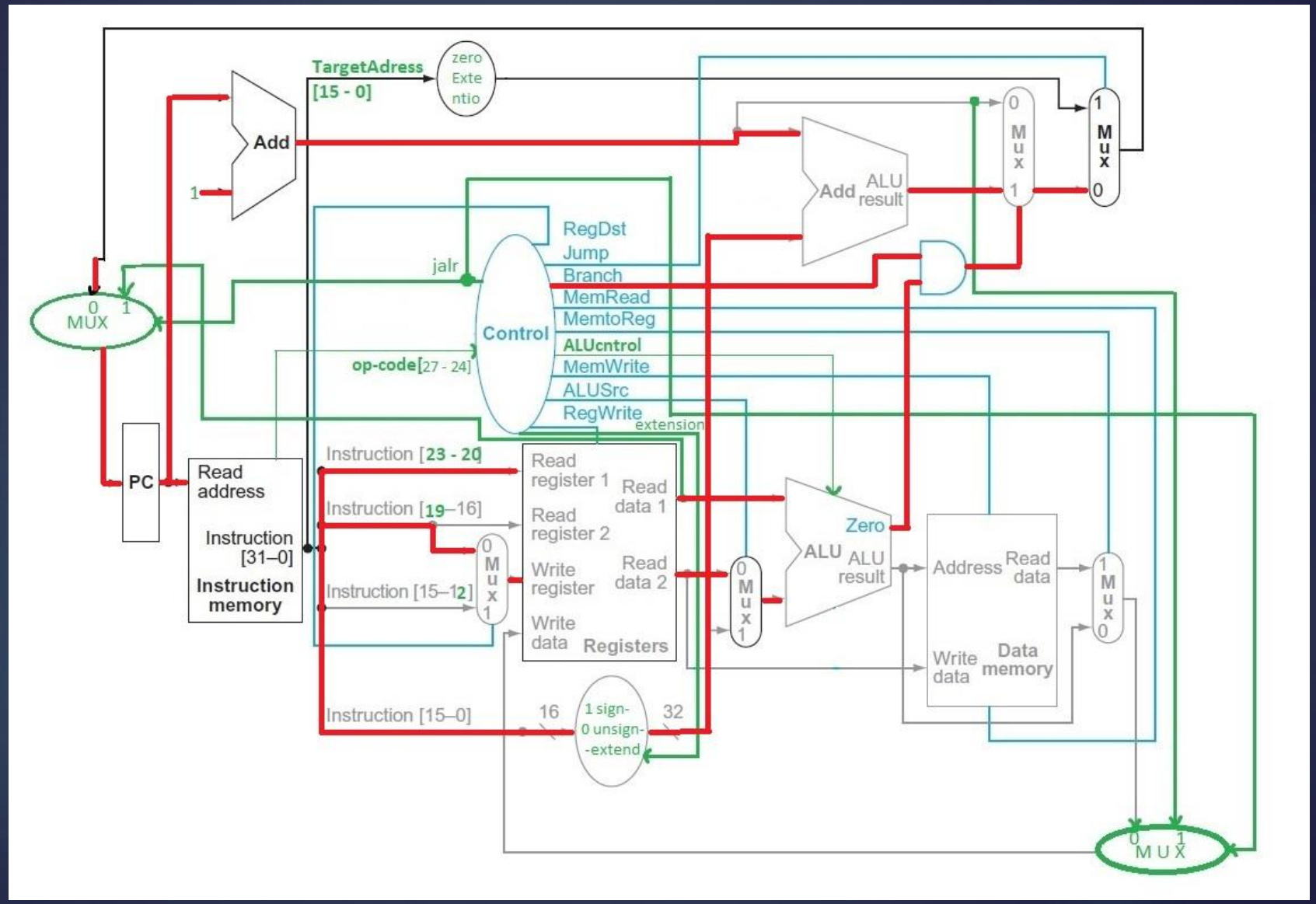
در این قسمت، مسیر یکی از دستورات معمولی I_Type با رنگ قرمز نشان داده شده است.

در این حالت، پی سی تغییر خاصی نمیکند و کلا +۱ می‌شود. همه سیگنال‌های کنترل که استفاده می‌شوند نیز علامت زده نشده است. با سیگنال RegWrite مقدار موردنظر (که توسط ALU بدست آمده است) در رجیستر مقصد نوشته می‌شود.



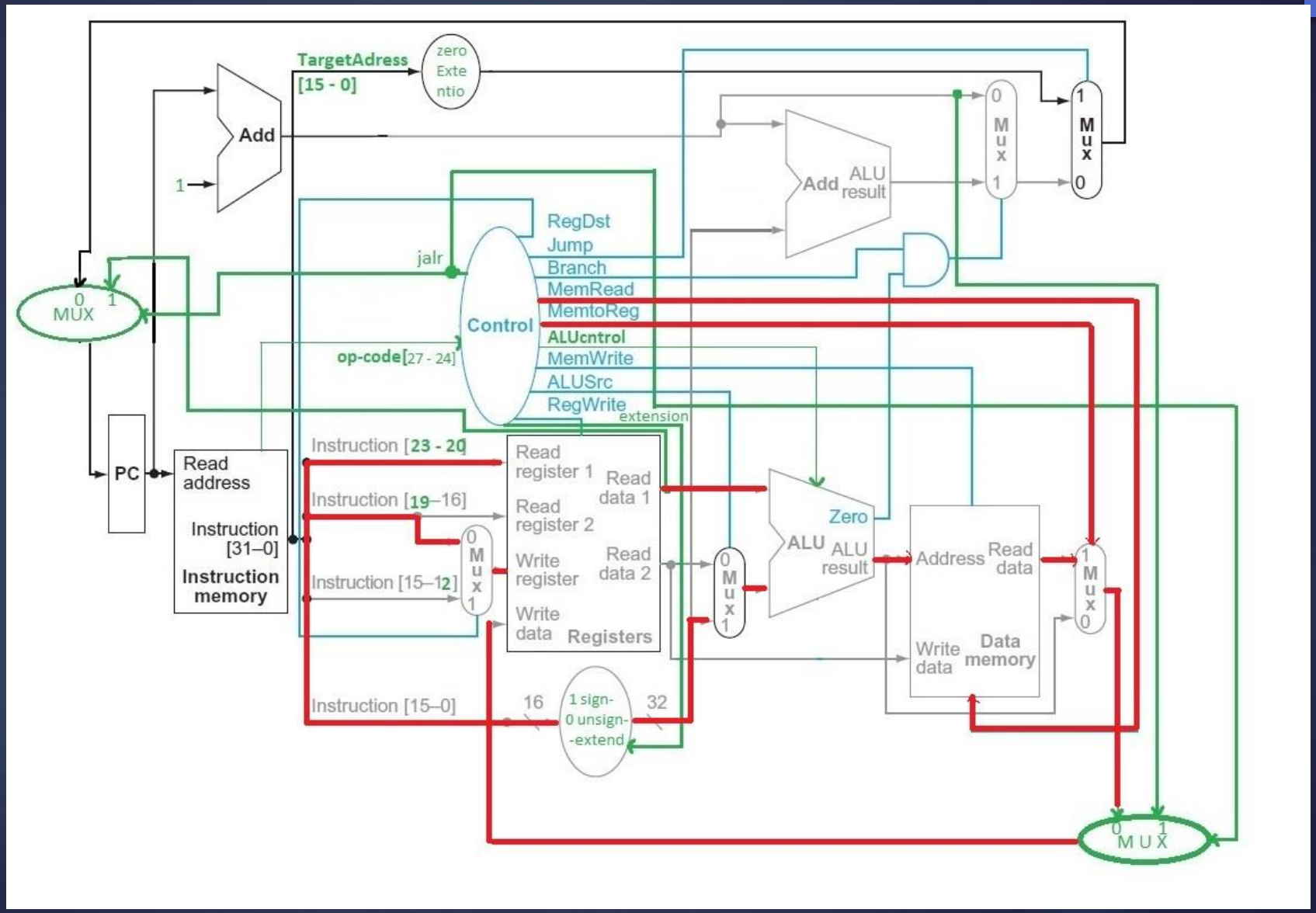
در این قسمت، مسیر یکی از دستورات I_Type، beq با رنگ قرمز نشان داده شده است.

در این حالت، فرض کرده‌ایم که دستور branch، taken باشد. درنتیجه دو رجیستری که مقدارشان خوانده می‌شود، مساوی است و سیگنال Zero از ALU خروجی true خواهد داشت. چون دستور نیز برنچ است، با توجه به آپ‌کد آن، سیگنال branch نیز &اش با Zero، true خواهد شد. در نهایت مقدار offset با pc+1 جمع شده و برای مقدار نهایی پی‌سی، انتخاب خواهد شد.



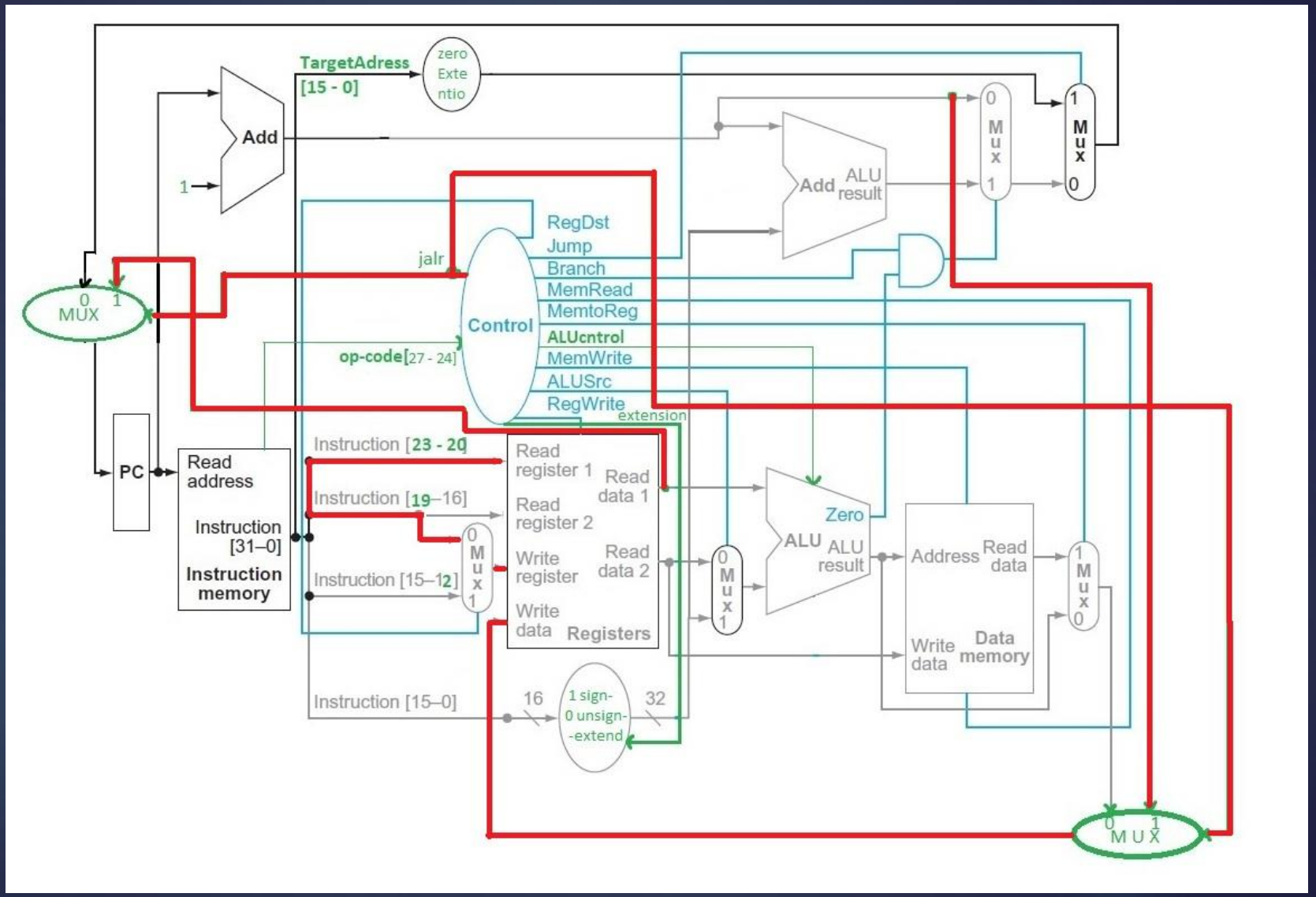
در این قسمت، مسیر یکی از دستورات I_Type، lw با رنگ قرمز نشان داده شده است.

در این حالت، خروجی ALU، آدرسی از موری است که داده‌اش را می‌خواهیم. خروجی ALU از جمع offset(بیت های ۰ تا ۱۵ – ۱۶ بیت کم ارزش از دستور ۳۲ بیتی) و رجیستر۱ بدست می‌آید. در نهایت، مقدار خوانده شده از موری در رجیستر مقصد ریخته می‌شود.



در این قسمت، مسیر یکی از دستورات I_Type، jalr با رنگ قرمز نشان داده شده است.

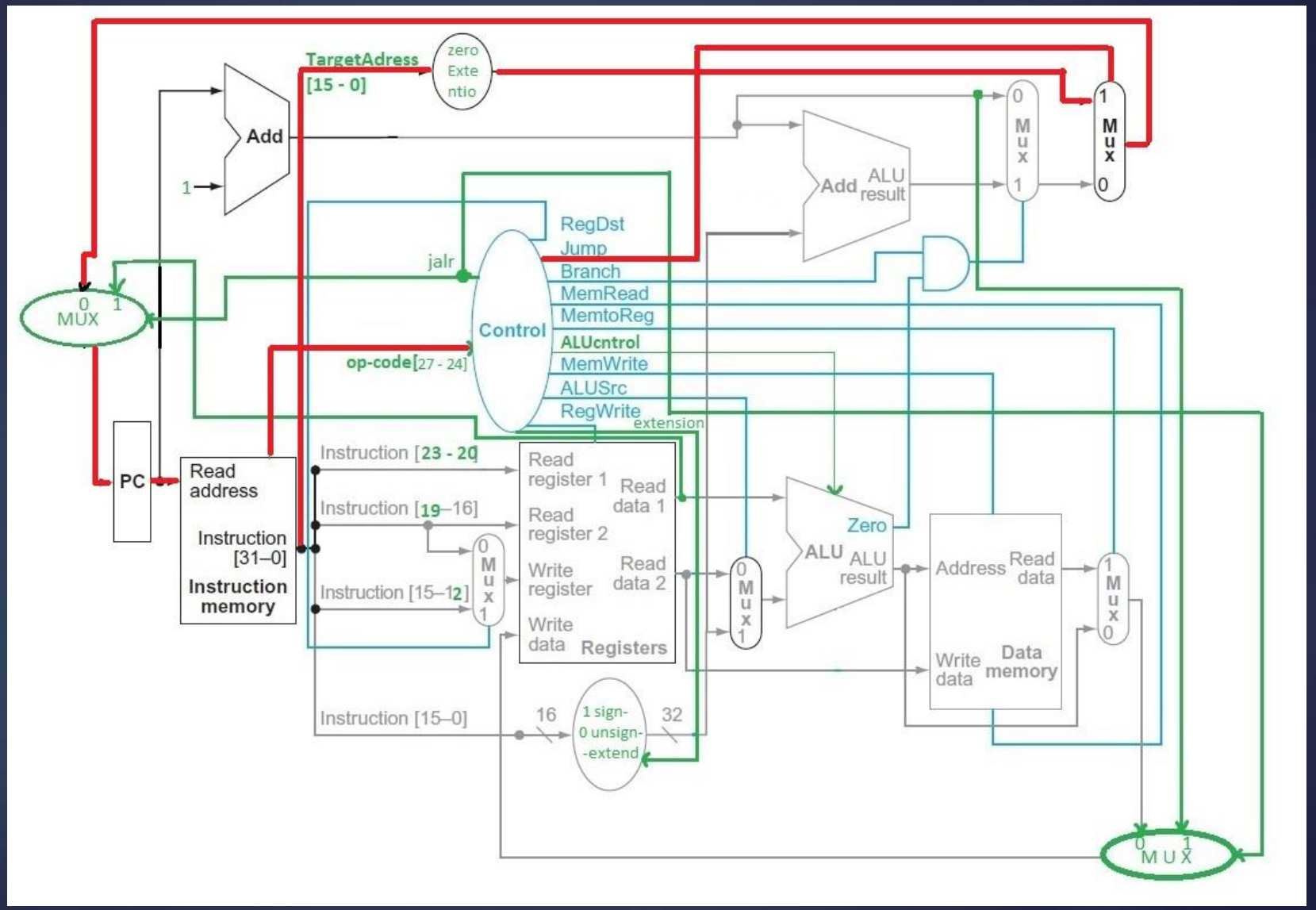
در این حالت، مقدار رجیستر۱ در پی‌سی ریخته می‌شود و مقدار pc+1 نیز در رجیستر۲، که در این نوع دستورات، رجیستر مقصد هستند ریخته می‌شود.



در این قسمت، مسیر یکی از دستورات J_Type، j با رنگ قرمز نشان داده شده است.

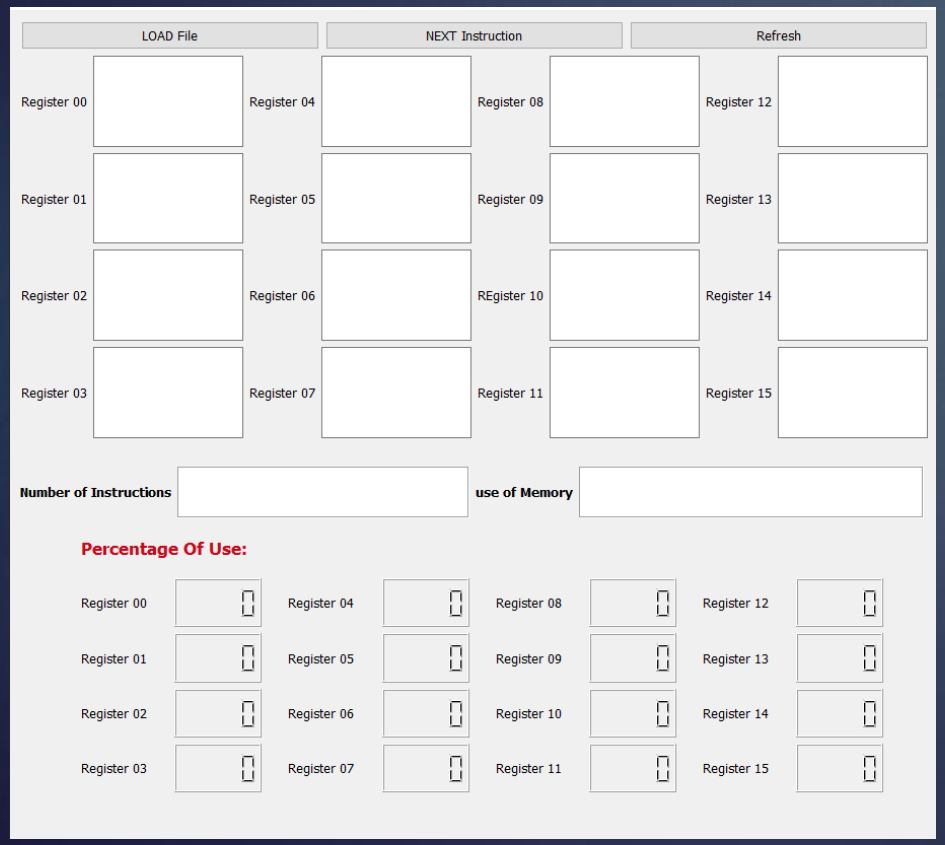
در این حالت، مقدار pc بعدی را TargetAdress این دستور مشخص می‌کند که توسط سیگنالی از کنترل که مشخص می‌کند دستور جامپ است، برای مقدار نهایی پی‌سی انتخاب می‌شود.

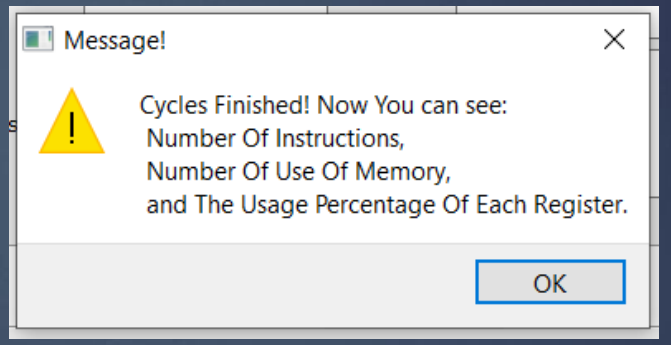
دستور دیگر J_Typeها، پردازنده تک‌سیکلی را متوقف می‌کند و در انتهای هربرنامه کد اسمبلی، آمده‌است.



Run and Execute

- هنگامی که برنامه اجرا می‌شود، با پنجره زیر روبه‌رو خواهید شد. با استفاده از دکمه Load می‌توانید فایل موردنظر خود را انتخاب کنید. هربار کلیک کردن روی دکمه next، دستور بعدی پردازش خواهد شد و در هر دور پردازش، مقدار رجیسترها در همان زمان نمایش داده خواهند شد. همچنین در انتهای برنامه پیامی مبنی بر تمام شدن آن و اینکه هم‌اکنون اطلاعاتی درباره برنامه چاپ می‌شود، دریافت خواهید کرد. در انتها، برای اینکه فایل جدیدی انتخاب کنید، باید مقادیر رجیسترها را Refresh کنید.





- نمایش درصد استفاده از رجیسترها تا ۴ رقم با معنی است.
درصورتی که بیش از ۴ رقم داشته باشیم، عدد گرد می‌شود.
- اگر از رجیستری استفاده نشده باشد ۰، و در صورتی که درصد استفاده کمتر از ۴ رقم با معنی باشد، ۰٫۰۰۰ نمایش داده می‌شود.