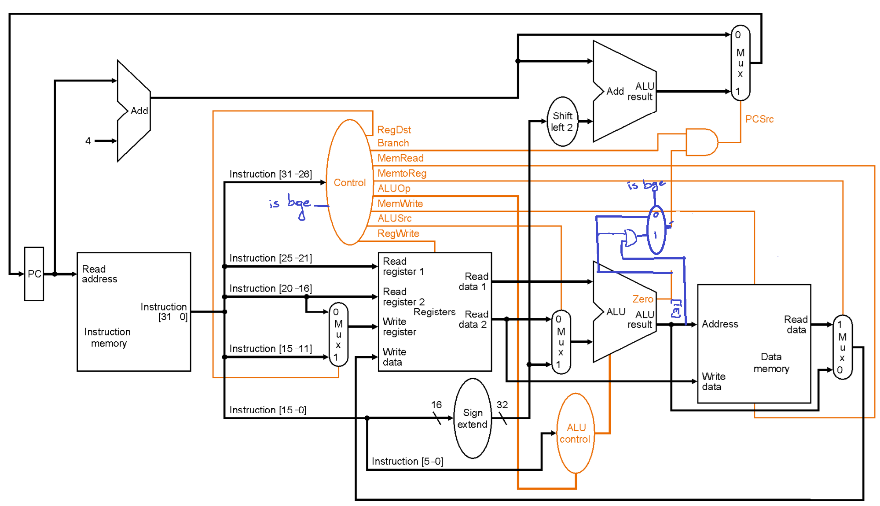
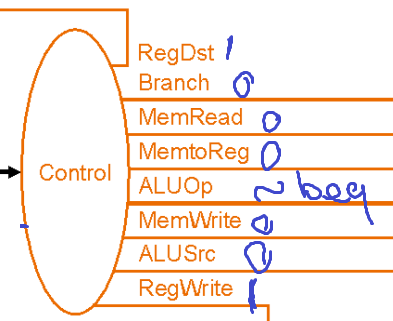
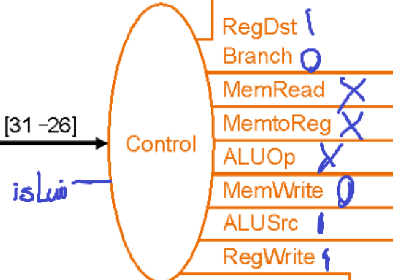
1. **(آ)**
2. همانطور که در شکل زیر مشاهده میکنید. یک سیگنال از control برای فهمیدن آنکه دستور bge وارد شده یا نه خارج میکنیم. سپس برای شرط زیرو در پرش یک مالتری پلکسر میذاریم که سلکت آن اگر صفر بود مقادر زیرو از آن خارج شود و در غیر این صورت باید مقدار OR زیرو و نات با ارزش ترین بیت خروجی Alu را خروجی میدهیم. چون درواقع اگر پرارزش ترین بیت خروجی که حاصل تفریق داده درون دو رجیستر ورودی است برابر 0 شود یعنی عدد ان مثبت است (حالت اورفلو رو درنظر نگرفتم.. درون صورت 33 بیتی فرضش کنین و باقی ماجرا) و داده اول تفریق بزرگ تر بوده است که معادل g است. همچنین تمام بقیه سیگنال های کنترلی Control به ویژه branch را مشابه حالت beq (یعنی همانsub ) تنظیم میکنیم.



1. مشابه قسمت اول از Control سیگنال خروجی isSlt رو میگیریم و به AluOp سیگنال حالت Branch یا sub را میدهیم تا برایمان rs و rt را از هم کم کند و مشابه قسمت قبل با 1 بودن بیت 31 ام متوجه منفی بودن حاصل تفریق و بزرگ تر بودن rt میشویم. درین صورت در ورودی کنترلی RegWrite رو بجای مستقیم وارد RF کردن، به یک گیت OR که یک سر آن همان سیگنال RegWrite است و وردی دیگرش And سیگنال isSlt و بیت 31 ام حاصل Alu ، متصل میکنیم. یعنی در صورتی که isSlt یا بیت 31 ام برابر 0 بود، همان RegWrite خروجی این مالتی پلکسر میباشد. همچنین ورودی Write data را نیز یک مالتی پلکسر میگذاریم که سیگنال ورودی آن همان And سیگنال isSlt و بیت 31 ام است. ورودی 0 آن همان سیم Data و ورودی 1 آن یک صفر 32بیتی است که میتوانیم با xor یک عدد 32 بیتی با خودش نیز بسازیم.



1. مشابه تصویر روبرو این خروجی سیگنال های کنترلی ما درین حالت است. حال ورودی write data را مشابه قبل از یک مالتی پلکسر عبور میدهیم که سیگنال سلکت آن برابر سیگنال isLui خارج شده از Control باشد و ورودی 1 آن نیز یا برابر یک اسپلیتر 32 بیتی به 2 تا 16 بیتی است که 16 بیت بالایی آن برابر همان 16 بیت اینستراکشن و 16 بیت دیگ 0 است و یا همون خروجی ساین اکستند شده 32 بیتی ساخته شده از 16 بیت اخر دستور وردی است که 16 بیت شیف چپ داه شده است.

**ب)** عه من همینطوریش در الف مشخص کردم... 😊

1. **(آ)** باید بیشترین زمان اجرای یک دستور از میان زمان هارا به عنوان زمان یک کلاک در نظر بگیریم زیرا تمام آنها باید بهرحال در 1 کلاک انجام شوند بنابر این جواب برابر 8 نانو ثانیه است.

**ب)** از آنجا که زمان اجرای هر دستور همان زمان کلاک است پس زمان اجرای کل این برنامه برابر 4539×8 یا همان 36312 نانو ثانیه است.

**ج**) برای هر دستور در نظر میگیریم که در هر کلاک پردازنده به ازای هر دستور چقدر از زمان صرف انجام کاری برای آن دستور نشده است و بعد حساب میکنیم در نهایت به طور متوسط چند ثانیه کاری انجام نشده:

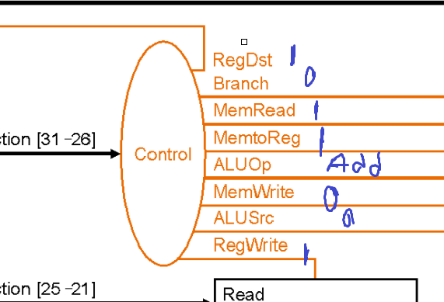
2 × 0.44 + 1 × 0.12 + 3 × 0.18 + 6 × 0.02 = 1.66𝑛𝑠

پس در هر 8 نانوثانیه 1.66 نانو ثانیه اش به طور میانگین اتلاف شده پس برای هر برنامه دلخواه به طور متوسط 20.75 درصد از زمان اتلاف میشود**.**

1. **آ)**
2. R\_type: I-Mem, Regs, Mux, Alu, Mux, Regs
3. Lw: Regs, Alu, D-Mem, Mux, Regs
4. Sw: I-Mem, Regs, Alu, D-Mem
5. Beq: I-Mem, Regs, Mux, Alu, Mux

**ب)** به ترتیب برابر 270 ، 410، 350 و 220 پیکو ثانیه میشود.

**ج)** مسیر بحرانی آن دستوری که طولانی ترین زمان را دارد. بنابرین جواب مسیر بحرانی lw است.

**د)** نرخ کلاک همان معکوس زمان هر کلاک است. بنابرین نرخ کلاک برابر 1 تقسیم بر 410 پیکو ثانیه یا به عبارتی برابر با 2.439 گیگا هرتز است

**4. آ)** این دستور نیازی به اضافه کردن سیگنال یا مالتی پلکسر جدید ندارد. و کافیست تنها مشابه شکل روبرو با توجه به آپکد این دستور، سیگنال های کنترلی را تنظیم کنیم. در این حالت مقدار درون دو رجیستر اول جمع شده و مقادر مجدود در آدرس حاصل جمع، در D-cache خوانده میشود و در نهایت در رجیستر مقصد ریخته میشود. البته توجه داشته باشید که کد عملیات Alu نیز باید معادل جمع دو عدد باشد...

**ب)** با توجه به انکه طول دستورات 32 بیتی است و اپکد 6 و نام رجیستر ها 5 بیتی، پس فرض میکینم افست 11 بیت آخر دستور است. ان را جدا میکنیم و با سیم ( 😊) با کمک بیت پر ارزش آن به یک عدد 32 بیتی sign extend اش میکنیم. ( البته برای صرفه جویی میتوان تا 16 بیت اول گسترشش داد و با یه مالتی پلکسر به خود گیت 16 به 32 مدار اصلی ان را ورودی داد که زحمت 16 بیت دیگر را بکشد). سپس آن مقدار 32 بیتی را به کمک Alu با دیتای درون rs جمع میزنیم سپس مقدار درون حافظه با آن آدرس را میخوانیم و با گیت های منطقی با مقدار موجود در rt (که از قبل Regs ما خروجی داده بود) یک جمع 32 بیتی میزنیم و خروجی آن را به ورودی 1 یک مالتیپلکسر با سلکتی برابر سیگنال کنترلی isAddm است و وردی صفرش همان سیم دیتا و خروجی آن به writing data Regs متصل است.

**5.** ابتدا مسیر بحرانی در هر دستور را مینویسیم:

1. R-Type: I-Mem , Regs , Mux , ALU , Mux , Regs

2. LOAD: I-Mem , Regs , ALU , D-Mem , Mux , Regs

3. STORE: I-Mem , Regs , ALU , D-Mem

4. Branch: I-Mem , Regs , Mux , ALU , Mux

5. Jump: I-Mem , Control , Mux

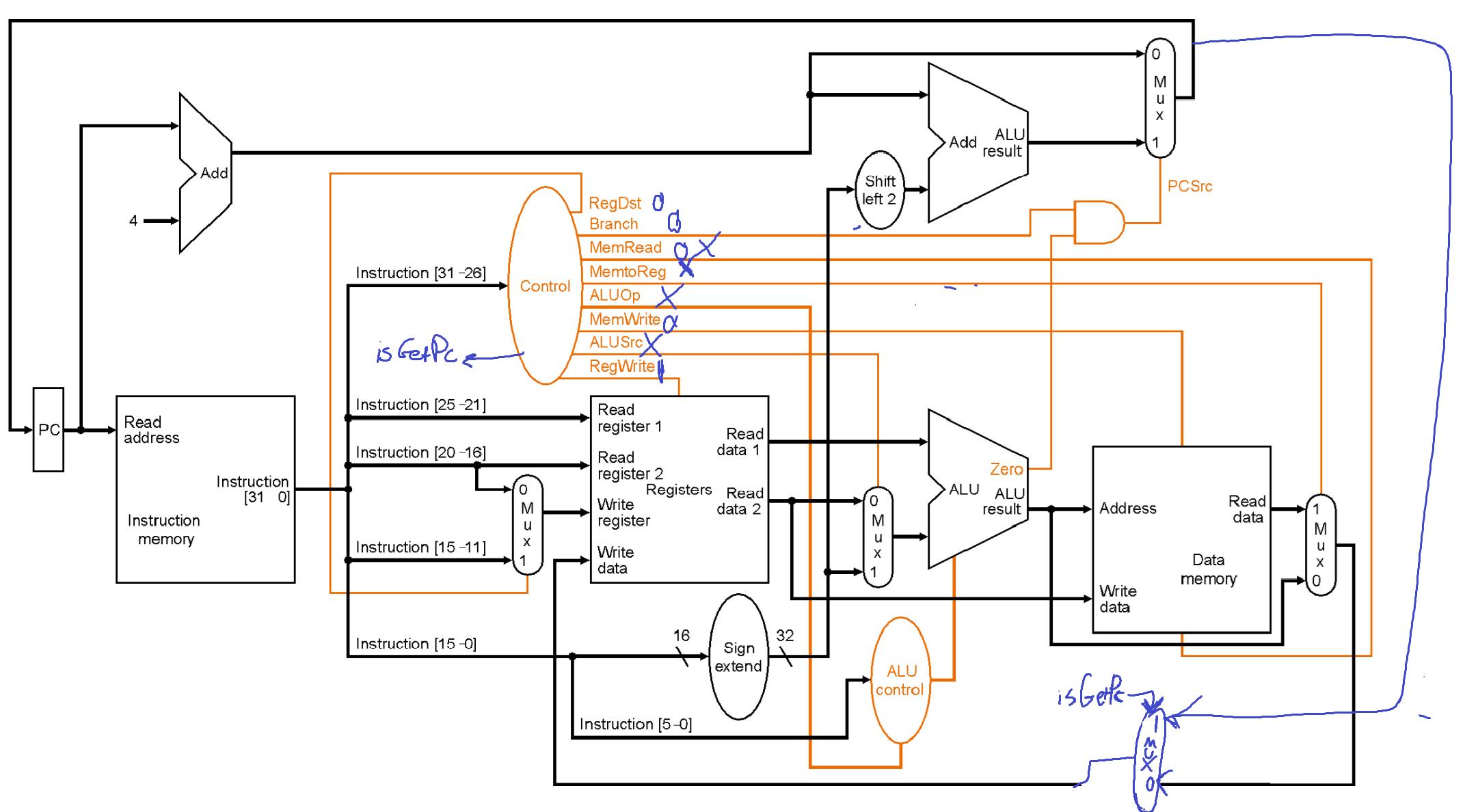
حال تاخیر هارا قبل و بعد از بهبود با توجه به مسیر ها محاسبه میکنیم.

قبل از بهبود، به ترتیب تاخیر ها برابر با 550، 830، 730، 470 و 300 پیکو ثانیه هستند.

و بعد از بهبود، به ترتیب تاخیر ها برابر است با 540، 775، 675، 460 و 315 پیکو ثانیه است

حال چون پردازنده single-cycle است زمان هر کلاک را برابر بیشینه مقدار های محاسبه شده، یعنی 830 و 775 در نظر میگیریم. تسریع نیز از حاصل تقسیم زمان قدیم به جدید محاسبه میشود، پس جواب برابر است

**6.** مشابه شکل یک مالتی پلکسر در مسیر دیتایی که برای نوشتن در رجسیستر ورودی داده میشود قرار میدهیم که سلکت آن سیگنال کنترلی isGetPc خارج شده از control است قرار میدهیم. همچنین باقی سیگنال های کنترلی درون شکل مشخص شده اند. پ.ن: نمیدونستم میخواین در رجیستر گفته شده PC این دستور رو ببنید یا بعدیش که PC+4عه.. بهرحال من درون شکل PC+4 رو به ورودی 1 مالتی پلکسرم وصل کردم.

****