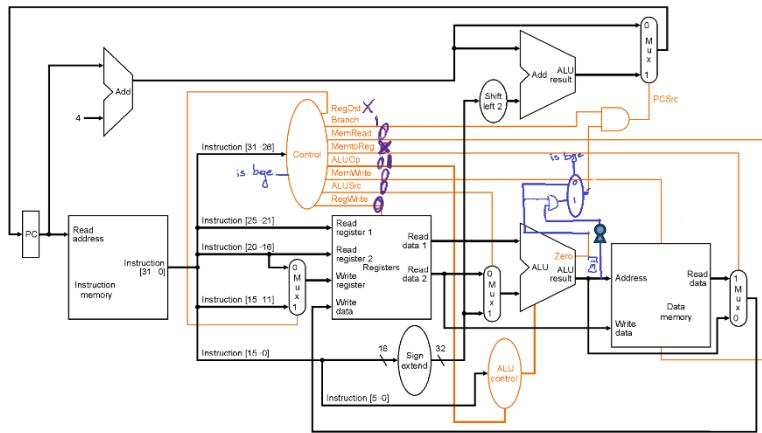
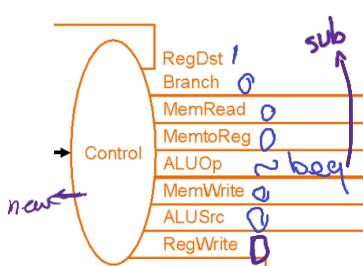


(۱) . ۱



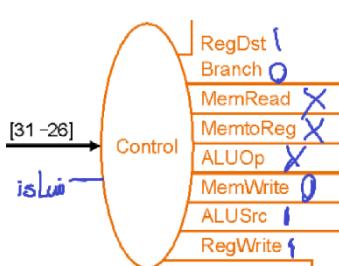
(۱) همانطور که در شکل زیر مشاهده میکنید. یک سیگنال از control برای فهمیدن آنکه دستور *bge* وارد شده یا نه خارج میکنیم. سپس برای شرط زیرو در پرش یک مالتیپلکسر میداریم که سلکت آن اگر صفر بود مقادیر زیرو از آن خارج شود و در غیر این صورت باید مقادیر *OR* زیرو و نات با ارزش ترین بیت خروجی *Alu* را خروجی میدهیم. چون در واقع اگر پرازش ترین بیت خروجی که حاصل تفربیق داده درون دو رجیستر ورودی است برابر ۰

شود یعنی عدد ان مثبت است (حالت اورفلو رو درنظر نگرفتم). درون صورت ۳۳ بیتی فرضش کنین و باقی ماجرا) و داده اول تفربیق بزرگ تر بوده است که معادل ۹ است. همچنین تمام بقیه سیگنال های کنترلی *Control* به ویژه *branch* را مشابه حالت *beq* (یعنی همان *sub*) تنظیم میکنیم.



(۲) مشابه قسمت اول از *Control* سیگنال خروجی *isSlt* را میگیریم و به سیگنال حالت *sub* یا *Branch* را میدهیم تا برایمان *rs* و *rt* را از هم کم کند و مشابه قسمت قبل با ۱ بودن بیت ۳۱ ام متوجه منفی بودن حاصل تفربیق 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 دیگر شیفت *RegWrite* سیگنال *isSlt* و بیت ۳۱ ام حاصل *Alu*، متصل میکنیم. یعنی در صورتی که *isSlt* یا

بیت ۳۱ ام برابر ۰ بود، همان *RegWrite* خروجی این مالتی پلکسر میباشد. همچنین ورودی *Write data* را نیز یک مالتی پلکسر میداریم که سیگنال ورودی آن همان *And* سیگنال *isSlt* و بیت ۳۱ ام است. ورودی ۰ آن همان سیم *Data* و ورودی ۱ آن یک صفر ۳۲ بیتی است که میتوانیم با *XOR* یک عدد ۳۲ بیتی با خودش نیز بسازیم.



(۳) مشابه تصویر روبرو این خروجی سیگنال کنترلی های کنترلی ما درین حالت است. حال ورودی را مشابه قبل از یک مالتی پلکسر عبور میدهیم که سیگنال سلکت آن برابر سیگنال *isLui* خارج شده از *Control* باشد و ورودی ۱ آن نیز یا برابر یک اسپلیت ۳۲ بیتی به ۲ تا ۱۶ بیتی است که ۱۶ بیت بالای آن برابر همان ۱۶ بیت اینستراکشن و ۱۶ بیت دیگ ۰ است و یا همون خروجی ساین اکستند شده ۳۲ بیتی ساخته شده از ۱۶ بیت اخر دستور وردی است که ۱۶ بیت شیف چپ داد شده است.

(ب) عه من همینطور بش در الف مشخص کردم... 😊



۲. (آ) باید بیشترین زمان اجرای یک دستور از میان زمان هارا به عنوان زمان یک کلاک در نظر بگیریم زیرا تمام آنها باید بهر حال در

۱ کلاک انجام شوند بنابر این جواب برابر ۸ نانو ثانیه است.

(ب) از آنجا که زمان اجرای هر دستور همان زمان کلاک است پس زمان اجرای کل این برنامه برابر $453.9 \times 8 = 3631.2$ نانو ثانیه است.

(ج) برای هر دستور در نظر میگیریم که در هر کلاک پردازندۀ 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:

$$2 \times 0.44 + 1 \times 0.12 + 3 \times 0.18 + 6 \times 0.02 = 1.66\text{ns}$$

پس در هر ۸ نانو ثانیه ۱.۶۶ نانو ثانیه اش به طور میانگین اتلاف شده پس برای هر برنامه دلخواه به طور متوسط ۲۰.۷۵ درصد از زمان اتلاف میشود.

(آ. ۳)

۱. R_type: I-Mem, Regs, Mux, Alu, Mux, Regs

۲. Lw: Regs, Alu, D-Mem, Mux, Regs

۳. Sw: I-Mem, Regs, Alu, D-Mem

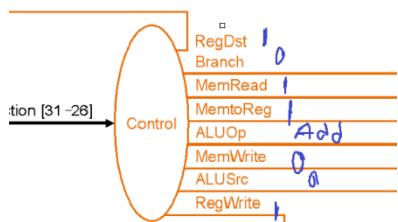
۴. Beq: I-Mem, Regs, Mux, Alu, Mux

(ب) به ترتیب برابر ۴۱۰، ۲۷۰، ۳۵۰ و ۲۲۰ پیکو ثانیه میشود.

(ج) مسیر بحرانی آن دستوری که طولانی ترین زمان را دارد. بنابرین جواب مسیر بحرانی ۱W است.

(د) نرخ کلاک همان معکوس زمان هر کلاک است. بنابرین نرخ کلاک برابر ۱ تقسیم بر ۴۱۰ پیکو ثانیه یا به عبارتی برابر با

۲.۴۳۹ گیگا هرتز است



۴. (آ) این دستور نیازی به اضافه کردن سیگنال یا مالتی پلکسر جدید ندارد. و کافیست تنها مشابه شکل رویرو با توجه به آپکد این دستور، سیگنال های کنترلی را تنظیم کنیم. در این حالت مقدار درون دو رجیستر اول جمع شده و مقدار 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 ۳۲ بیتی است و اپکد ۶ و نام رجیستر ها ۵ بیتی، پس فرض میکنیم افست ۱۱ بیت آخر دستور است.

آن را جدا میکنیم و با سیم (??) با کمک بیت پر ارزش آن به یک عدد ۳۲ بیتی sign extend کنیم. (البته برای صرفه جویی میتوان تا ۱۶ بیت اول گسترشش داد و با یه مالتی پلکسر به خود گیت ۱۶ به ۳۲ مدار اصلی ان را ورودی داد که زحمت ۱۶ بیت دیگر را بکشد). سپس آن مقدار ۳۲ بیتی را به کمک Alu با دیتای درون ۷S 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 ۳۲ بیتی میزنیم و خروجی آن را به ورودی ۱ یک مالتیپلکسر با سلکتی برابر سیگنال کنترلی isAddm است و وردی صفرش همان سیم دیتا و خروجی آن به writing data Regs متصل است.

۵. 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، 470، 730، 830 و 300 پیکو ثانیه هستند.

و بعد از بهبود، به ترتیب تاخیر ها برابر است با 540، 575، 775، 460 و 460 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.

تسريع نیز از حاصل تقسیم زمان قدیم به جدید محاسبه میشود، پس جواب برابر $\frac{0.83}{0.775} = 1.07$ است

۶. مشابه شکل یک مالتی پلکسر در مسیر دیتایی که برای نوشتن در رجیستر 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 بینید یا بعدیش که $PC+4$ عده. به حال من درون شکل ۱ مالتی پلکسرم وصل کردم.

