بسمه تعالی

گزارش کار پروژه معماری کامپیوتر

دانشگاه صنعتی شریف

استاد: دکتر سربازی

اعضای گروه:

سیدمحمدرضا خسرویان

محمدرضا بدری

سام خانکی

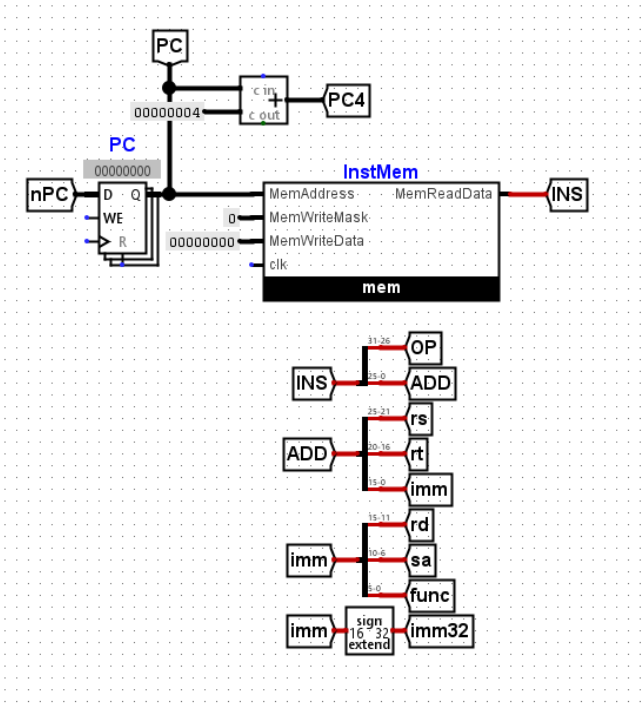
فاز اول پروژه

خرداد ۱۴۰۲

در ابتدا cpu را به بخش های زیر تقسیم میکنیم:

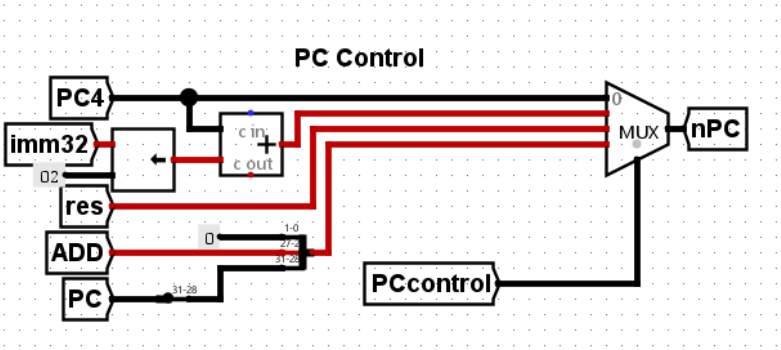
Instruction, PC Control, register file, ALU, ALU controller, Clock, Memory

در بخش instruction، رجیستر PC برای کنترل آدرس قرار داده شده است و در هر مرحله دستورات را برای راحتی بخش بندی میکنیم:



دستور ۳۲ بیتی INS به قسمت های OP, ADD, RS, RT, RD, SA, Func, Imm تقسیم بندی میشود و سپس imm را sign extend میکنیم و با این کار تمام اجزای کد را در tunnel های متناظر داریم.

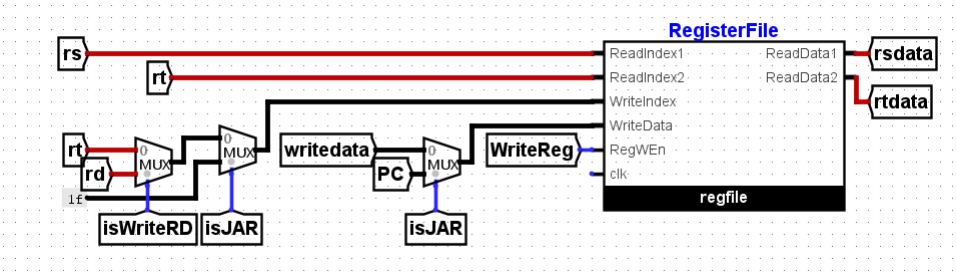
در بخش Control PC، PC بعدی را با توجه به کد و حالت بندی های مختلف ایجاد میکنیم:



۴ حالت مختلف برای PC بعدی وجود دارد، یا به خط بعدی میرویم مانند اکثر کد ها که در این حالت nPC = PC + 4، یا دستورات branching مانند bne رخ میدهند و آدرس جدید با توجه به مقدار imm<<2 ایجاد میشود و این مقدار با PC+4 جمع میشود، یا PC مستقیم از روی رجیستر با دستور jr خوانده میشود و یا به طور مستقیم با دستورات j و jar، PC جدید بر اساس ADD یافت میشود.

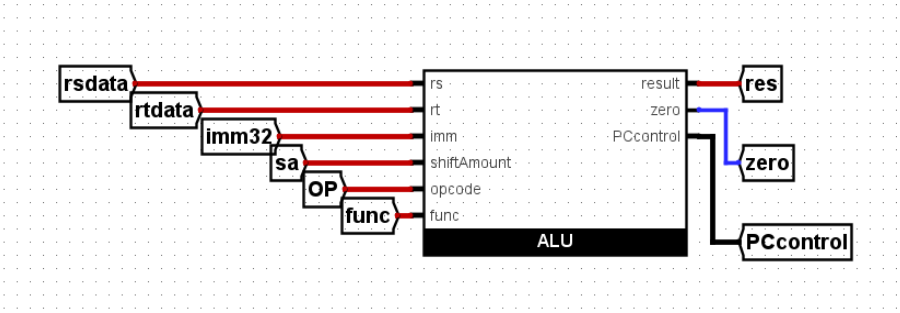
همچنین PCcontrol بین این ۴ حالت بر اساس نوع کد خروجی را تعیین میکند.

در بخش register file، دیتا های موجود در رجیستر های کد را از رجیسترفایل میخوانیم و در صورت نیاز جواب ها و حاصل ها را در آن write میکنیم:



دو ورودی اول آدرس های rt و rs هستند که در قسمت دستورات استخراج کرده ایم و محتوای این دو رجسیتر خروجی رجیستر فایل است. در قسمت write، نوشتن با writeReg کنترل میشود، همچنین سه نوع نوشتن داریم، یا از دستور jar استفاده شده که در این صورت رجیستر ra یا همان رجیستر 31 باید محتوای PC+4 را ذخیره کند، یا دستور از نوع R هست و محتوا باید در رجیستر RD ذخیره شود و در حالت سوم دستور از نوع I هست و محتوا باید در رجیستر RT ذخیره شود. این حالات را با isJAR و isWriteRD تعیین میکنیم.

بخش ALU شامل دو بخش هست، ورودی خروجی ALU و خود ALU که در فایل au بخش ALU قرار دارد:

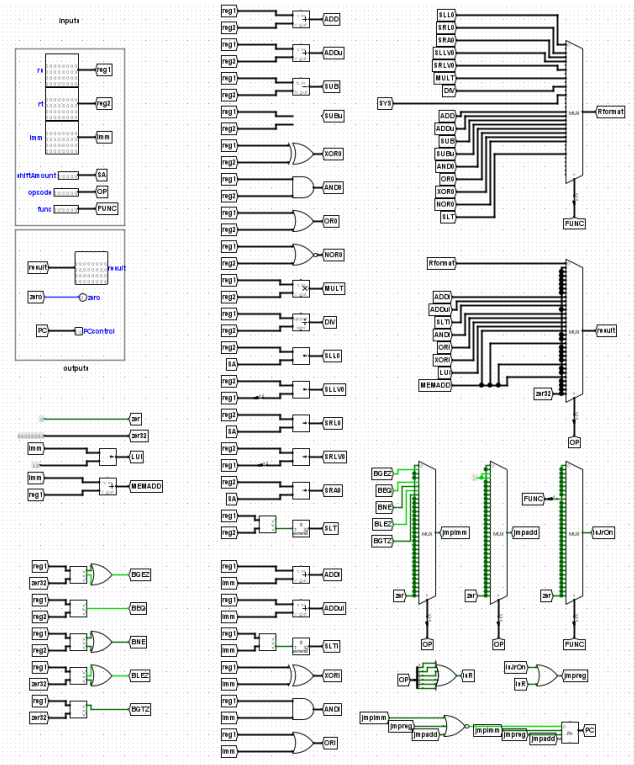


قطعه های مورد استفاده کد در بخش INS و خروجی های رجیستر فایل را به عنوان ورودی به ALU میدهیم و خروجی PC کنترل برای تعیین نوع جامپ به بخش مربوطه میرود.

در داخل ALU ابتدا با توجه به ورودی ها (در قسمت چپ بالا ورودی و خروجی ها تعیین شده اند)، تمام خروجی های ممکن را ایجاد میکنیم (قسمت وسط و سمت چپ پایین ورودی / خروجی ها) و سپس با توجه به محتوای کد، خروجی درست را به result متصل میکنیم. برای این کار از mux استفاده میشود به طوری که ورودی آن ۵ بیت از opcode و func هستند و با استفاده از همین بیت های select خروجی تعیین میشود.

دو mux، یکی برای حالت rFormat و دیگری برای حالت iFormat در ALU قرار داده شده اند که در یکی بیت سلکت از func و در دیگر از OP گرفته میشود و خروجی rFormat به عنوان حالتی که OP = 000000 در MUX دوم داده شده است و در نهایت خروجی MUX دوم به result ALU متصل میشود.

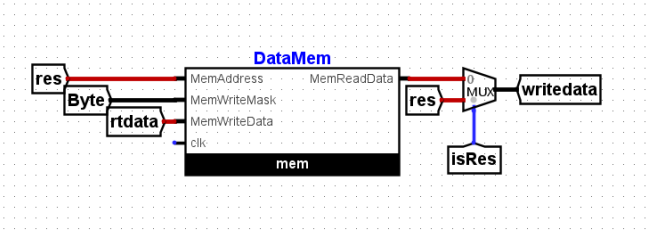
همچنین بر اساس نوع کد و نوع پرش PCcontrol نیز تعیین میشود (در قسمت پایین راست) :



در بخش ALU control صرفا تک بیتی های انتخاب که تا به حال با آنها برخورد کرده ایم مانند isJar و writeReg را با توجه به نوع کد تعیین میکنیم و همچنین میتوان بخش کنترل PC از ALU را به کنترل ALU در صورت نیاز منتقل کرد. این بخش با نام ALU_Control در فایل au قابل دسترسی میباشد.

در بخش CLK نیز به ترتیب INS، MEM، REG کلاک فعال میخورند و سپس دوباره INS کلاک میخورد و هر کلاک ALU شامل کلاک خوردن به ترتیب این سه بخش و سه کلاک است.

در نهایت در بخش MEMORY یک مموری دیتا داریم که در کد های sb, sw,lb, lw از آن استفاده میشود:



همواره ورودی آدرس از ALU result تعیین میشود و دیتای مورد نیاز برای نوشتن در مموری همواره در RT قرار میگیرد. قسمت ۴ بایتی Byte در واقع تعیین کننده بیت هایی هست که اطلاعات در آنها نوشته میشوند که در دستور sb استفاده میشود و توسط ALU Control محتوای آن تعیین میشود.

همچنین در قسمت خروجی، محتوای به tunnel، writedata منتقل میشود که این تانل در register file استفاده میشود، دیتایی که در رجیستر نوشته میشد، یا حاصل کارهایی روی متغیر ها میباشد که در ALU انجام شده و در result قرار دارد، یا قسمتی از حافظه است که خروجی حافظه است و در دستورات lw و lb از آن استفاده میشود و isRes که در ALU Controller تعیین میشود، بین این دو حالت انتخاب میکند.