توضیح نحوه کار Pipline:

۱) اولین قسمت از مرحله multiplexer قرار دارد که بین مقدار قبلی PC به علاوه ۴ و یا مقدار آمده از branch به وسیله PCSrc انتخاب می کند اگر صفر باشد مقدار قبلی و اگر یک باشد مقدار branch را برمی گیرد.

ورودی ها و خروجی ها ۶۴ بیتی هستند.

موضوع: _____ تاریخ: / /

(۲) PC قرار دارد که مقدار خروجی multiplexer قسمت (۱) را در خود ذخیره می کند و در طول بعدی آن را به مرحله بعد می دهد. در ابتدا هم مقدار داخل PC صفر است که یعنی برنامه از خانه صفر شروع می شود. (خانه صفر instuction memory در قسمت (۴)) ورودی و خروجی ۶۴ بیتی هستند.

(۳) در قسمت بعدی adder قرار دارد که مقدار خروجی از PC قسمت (۲) را دریافت کرده و آن را به علاوه ۴ کرده و آن را به عنوان ورودی multiplexer قسمت (۱) می دهد. پاسخ

ورودی و خروجی ۶۴ بیتی هستند.

(۴) در instruction memory چون فقط از آن خوانده می شود و در آن نوشته نمی شود به clock احتیاجی نیست.

در این قسمت داده های دستورات برنامه ذخیره می شوند. (در اینجا دستورات داده شده به طور پیش فرض در این قسمت ذخیره شده اند) (دستورات ۳۲ بیتی هستند)

و با توجه به خروجی PC داده که آدرس آن PC است از آدرس این قسمت خوانده شده و به مرحله بعد داده می شود.

ورودی ۶۴ بیتی و خروجی ۳۲ بیتی است.

(۵) رجیستر IF/ID قرار دارد که در خود باهر کلاک مقدارهای خروجی PC قسمت (۲) و instuction memory قسمت (۴) را در خود ذخیره می کند تا مقادیر فقط بتوانند در کلاک ها تغییر کنند و از مقادیر غیر درست بین کلاک ها استفاده نشود. در این قسمت ورودی و خروجی.

ورودی و خروجی ۹۶ بیتی است.

موضوع: _____ تاریخ: / /

④ کنترل در این مسئله قرار دارد که datapath را به نوعی مدیریت می‌کند. مقدار ورودی آن بیت‌های مربوط به خروجی ④ instruction memory در خروجی ⑤ IF/ID می‌باشد. و مقادیر کلیدی branch، mem_read، mem_write، mem_to_reg، alu_op، alu_src و reg_write را خروجی می‌دهد. (باتوجه جداول کتاب)

این مقادیر با استفاده از مقدار ذخیره شده در opcode دستورات یا ۷ بیت اول خروجی instruction memory ها تعیین می‌شود.

باتوجه به اینکه دستورات محدود شده بودند به نوع R، branch، store و load این دستورات را می‌توان بیت‌های ۴ و ۵ و ۶ در opcode از هم تشخیص داد. اگر بیت ۵ و ۶ هردو ۱ بودند یعنی دستور branch است در غیر این صورت اگر بیت ۴ و ۵ هردو ۰ بودند دستور load است و در غیر این صورت اگر بیت ۴، ۰ و بیت ۵، ۱ بود یعنی دستور store است و در غیر این صورت دستور R_type است.

⑦ Registe memory قرار دارد که مقادیر داخل هر رجیستر را مدیریت می‌کند که ۲ ورودی می‌گیرد اول و دوم شماره رجیستر هایی باشند که مقدار ۵ بیتی است که مربوط به بیت‌های ۱۵ تا ۱۹ و ۲۰ تا ۲۴ در بیت‌های مربوط به خروجی ④ instruction memory در خروجی ⑤ IF/ID می‌باشد.

و ورودی های بعدی برای write کردن داده در آن می‌باشد که اول باید Reg_write آن ۱ شود که خروجی کنترل ⑥ است که وابسته به دستورات مختلف است.

موضوع: تاریخ: / /

۶۴ بیتی ۵ بیتی
اول write register دوم write data که مقادیر آن از رجیستر MEM,WB
و multiplexer قسمت می‌آیند. و دو خروجی آن هم ۶۴ بیتی هستند.

(A) قسمت imm Gen مخصوص تولید immediate می‌باشد که ورودی آن ~~[illegible]~~
بیت‌های مربوط به خروجی instuction memory (۴) در خروجی IF/ID (۵) می‌باشد.

در این قسمت با توجه به ۷ بیت اول instuction ، یا همان opcode دستور از نوع
store ، load و یا branch انتخاب می‌شود. و مقدار immediate مخصوص آن
دستور ساخته می‌شود.

store → بیت‌های ۷ تا ۱۱ + بیت‌های ۲۵ تا ۳۱
load → بیت‌های ۲۰ تا ۳۱
branch → بیت‌های ۸ تا ۱۱ + بیت‌های ۲۵ تا ۳۰ + بیت ۷ + بیت ۳۱

ورودی این قسمت ۳۲ بیتی و خروجی ۶۴ بیتی است.

موضوع: _____ تاریخ: / /

(۹) رجیستر ID/EX در خود با هر کلاک داده های خروجی کنترل قسمت های (WB و M و EX) و بیت‌های مربوط به خروجی ① PC در خروجی ② IF/ID و دو خروجی ۶۴ بیتی در ⑦ Register memory ، خروجی ۶۴ بیتی ⑧ immgen و بیت‌های ۱۲ تا ۱۴ و ۳۰ خروجی ④ instuction memory در خروجی ⑤ IF/ID و همچنین بیت های مربوط به ۷ تا ۱۱ دستور خود نقد که مربوط به write کردن در رجیستر است (شماره مربوط به رجیستر یا write register) را در خود ذخیره می‌کند.

(۱۰) multiplexer قرار دارد بین بیت های مربوط به خروجی ⑦ registermemory و در خروجی ID/EX و (۶۴ بیتی) و بیت‌های مربوط به خروجی ۶۴ بیتی ⑧ immgen با توجه به ALUsrc انتخاب می‌کند.

(۱۱) در Alu control مقادیر مورد نیاز برای انتخاب عملیات مورد نیاز در Alu انتخاب می‌شود ورودی این قسمت بیت‌های ۱۲ تا ۱۴ و ۳۰ مربوط به دستور در خروجی ⑨ ID/EX است و همچنین Alu-op (۲ بیتی) به خروجی ⑨ ID/EX ⑩ بیت مربوط به خروجی کنترل است. با توجه به این مقادیر و جداول داخل کتاب خروجی مورد نظر که ۴ بیت بوده و عملیات مورد نظر Alu می باشد تولید می‌شود.

(۱۲) در Alu ورودی اول ۶۴ بیت بیت‌های مربوط به خروجی اول Register memory (۷) در خروجی ID/EX (۹) می‌باشد و همچنین خروجی ۶۴ بیتی قسمت multiplexer (۱۰) می‌باشد و همچنین خروجی Alu control که عملیات مورد نظر را تعیین می‌کند

00000 → and
0001 → or
0010 → add
0110 → sub

خروجی آن هم حاصل این عملیات است و همچنین یک flag z و یک flag C است که برای flag z دستور beq در branch مثل مساوی بودن مقدار را اطلاع می‌دهد.

(۱۳) یک adder که مقدارهای بیت‌های مربوط به خروجی PC (۲) در خروجی ID/EX (۹) و بیت‌های مربوط به خروجی ۶۴ بیتی immGen (۸) در خروجی ID/EX (۹) ۶۴ بیتی را جمع می‌زند خروجی ۶۴ بیتی این adder به رجیستر EX/MEM (۱۴) می‌رود و با کلاک هم به multiplexer قسمت (۱) می‌رود.

(۱۴) رجیستر EX/MEM در خود با هر کلاک داده‌های خروجی کنترل قسمت‌های (WB و M) و خروجی قسمت adder (۱۳) و flag z و خروجی Alu (۱۲) و خروجی دوم بیت‌های مربوط به Register memory (۷) در خروجی ID/EX (۹) و همچنین write registe از خروجی ID/EX (۹) را در خود ذخیره می‌کند.

موضوع: تاریخ: / /

(۱۵) یک and gate که مقدار flagz و branch را باهم and می‌کند و خروجی آن هم به عنوان Pcscr به multiplexer قسمت (۱) می‌رود.

(۱۶) Datamemory یا داده‌ها [illegible] خوانده یا نوشته می‌شوند و یک ورودی اول آن خروجی مربوط به نتیجه Alu است که برای load و store کاربرد دارد. ورودی اول آن خروجی مربوط به نتیجه Alu است که به عنوان آدرس استفاده شده و ورودی دوم آن هم بیت‌های مربوط به خروجی دوم در Register memory (۷) است که هر دو از خروجی EX/MEM (۱۴) می‌آیند. همچنین ۲ سیگنال MEMwrite و MEM Read هم برای این قسمت وارد می‌شوند که خروجی کنترل هستند که از خروجی EX/MEM (۱۴) می‌آیند.

(۱۷) رجیستر MEM/WB با هر کدام داده‌های مربوط به خروجی کنترل قسمت WB و خروجی Datamemory (۱۶) و بیت‌های مربوط به خروجی دوم Registermemory (۷) در خروجی EX/MEM (۱۴) و همچنین داده‌های مربوط به write Register در خروجی EX/MEM (۱۴) در آن ذخیره می‌شوند.

(۱۸) یک multiplexer که بین بیت‌های مربوط به خروجی alu (۱۲) و datamemory (۱۶) خروجی [illegible] با توجه به MEM/WB (۱۷) MEMtoReg که خروجی کنترل است انتخاب کرده و همراه با

موضوع:
تاریخ: / /

بیت‌های مربوط به write register از خروجی MEM/WB به registermemory برمی‌گردد