1. آ) از استفاده میکنیم. پس:

ب) دو برابر شدن فرکانس، مدت زمان هر چرخه کلاک را نصف می‌کند، اما تعداد چرخه‌های مورد نیاز برای هر دستور (CPI) را تغییر نمی‌دهد. بنابراین، CPI مستقل از فرکانس است. اما افزایش تعداد کلاک ها به ازای هر دستور، CPI را بیشتر میکند. بنابراین همان فرمول قبل را داریم که چهار واحد افزایش یافته:

ج) میدانیم زمان اجرای یک برنامه در یک پردازنده از فرمول محاسبه میشود. بنابراین:

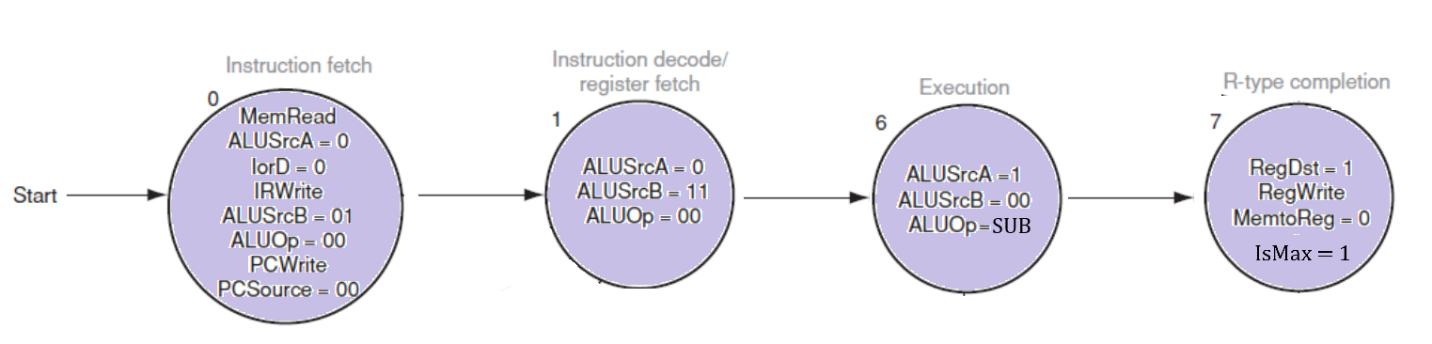
*بنابراین پردازنده دوم بهتر است و میزان تسریع ان برابر است با:*

*د) برای تصمیم گیری صرفا CPI رو در هر یک از حالت های گفته شده محاسبه میکنیم:*

*همانطور که میکنید مقدار اولی کمتر از پس بهبود بهتری است.*

1. *آ) در اجرای این دستور به ALU دستور sub را میدهیم و از خروجی ALU که rs-rt است استفاده میکنیم. اگر حالت overflow را در نظر نگیریم، بیت آخر (31 ام) نشاندهنده علامت این تفریق است؛ اگر این بیت 1 باشد یعنی حاصل خروجی منفی بوده و مقدار rt بیشتر از rs است و در بقیه حالات این بیت 0 است. البتدا یک سیگنال کنترلی جدید به نام IsMax از ALU خارج میکنیم که تنها در صورتی که دستور ما max باشد 1 میشود.در ادامه کافی است از دو مالتی پلکسر جدید در راه ورودی WriteData استفاده کنیم.سلکت مالتی پلکسر اول سیگنال IsMax است و ورودی 0 آن سیمی است که اکنون به WriteData وصل است و خروجی آن را به WriteData وصل میکنیم؛ ورودی 1 آن برابر خروجی مالتی پلکسر دوم است. سلکت مالتی پلکسر دوم همان بیت آخر خروجی ALU که در رجیستر ذخیره شده است. به ورودی 1 آن مقدار rt(Read Data 2) و به ورودی 0 آن rs(ReadData 1) را وصل میکنیم.*

*ب) این دستور غالبا شبیه دستورات R\_Type است. در IF دستورات خوانده میشود. در ID دستور کدگشایی میشود و مقادیر rs, rt خوانده میشوند. در مرحله EX دستور SUB روی ورودی ها اجرا میشود و خروجی ALU در ALUOut ذخیره میشود. مرحله MEM را مشابه سایر دستورات R\_Type نداریم و در مرحله WD هم مقدار Write Data توسط مالتی پلکسر هایی که اضافه کردیم مشخص میشود و سپس در rd نوشته میشود. سیگنال های کنترلی را نیز در شکل کشیدم:*

**

*ج) در MIPS، دستورla مخفف “load address” است و در واقع یک* ***pseudo-instruction*** *(دستور ساختگی) است که برای بارگذاری آدرس یک برچسب (label) در ثبات به کار می‌رود و اسمبلر آن را به مجموعه‌ای از دستورات واقعی (معمولاً lui و سپسori یا addi) تبدیل می‌کند. پس این دستور عملا 4+4=8 کلاک طول میکشد. برای سایر دستورات نیز تعداد کلاک ها مشابه اسلاید هاست. بنابراین داریم:*

1. *آ) برای Single-cycle ابتدا زمان مسیر بحرانی هر یک از دستورات رو پیدا میکنیم:*

|  |  |
| --- | --- |
| *Time* | *Intruction* |
| *200+50+100+50 = 400* | *R-Type* |
| *200+50+100+200+50=600* | *lw* |
| *200+50+100+200=550* | *sw* |
| *200+50+100=350* | *beq* |
| *200* | *J* |

*مثل همیشه lw بیشترین زمان ررا میبرد که 600 واحد زمانیست. پس در Single-cycle تمام دستورات در یک کلاک 600 واحدی اجرا میشوند.*

*در multi-cycle نیز ابتدا باید بیشترین زمان را بین 5 مرحله داده شده پیدا کنیم که همانطور که مشاهده میکنید این زمان متعلق به Mem با 200 واحد زمانیست. بنابرین تمام کلاک ها 200 واحد زمانی طول میکشند و داریم:*

|  |  |  |
| --- | --- | --- |
| *Time* | *Cycle count* | *Intruction* |
| *2004 = 800* | *4* | *R-Type* |
| *2005=1000* | *5* | *lw* |
| *2004 = 800* | *4* | *sw* |
| *2003 = 600* | *3* | *beq* |
| *2003= 600* | *3* | *J* |

*ب) میدانیم در سینگل سایکلCPI برابر 1 است. برای multi نیز از فرمول استفاده میکنیم:*

*ج) این تسریع را با فرمول همیشگی محاسبه میکنیم:*

*هیچی دیگه مشاهده میکنیم که تسریع نداریم.*

1. *خب بیاین فرض کنیم در برنامه a% از دستورات از نوع X، b% از دستورات از نوع Y و c% از دستورات از نوع Z هستند. حال CPI هر کدام از پردازنده ها در این برنامه را با همان فرمول همیشگی محاسبه میکنیم.*

*حال از فرمول و داده هایی که برای تسریع داشتیم استفاده میکنیم:*

*چون جمع 3 متغیر ما برابر 1 است، داریم: a = b = c = 0.33 و برنامه به نسبت یکسان از هر نوع دارد! 😊*

*برای محاسبه کران بالا حس میکنم میشه تخمین بهتری زد ولی فعلا فقط همین به ذهنم میرسه:*

*آ) برای نمایش 12 تا FF به 4 بیت نیاز داریم. پس ورودی هایROM ما بیت های OpCode فعالی و FF فعلی است و خروجی های آن FF بعدی و سیگنال های کنترلی است.حال سراغ محاسبه میرویم.*

*ب) از ROM اول برای ایجاد سیگنال های کنترلی (30 بیت) استفاده میکنیم. برای اینکار تنها به حالت FF در ورودی نیاز داریم:*

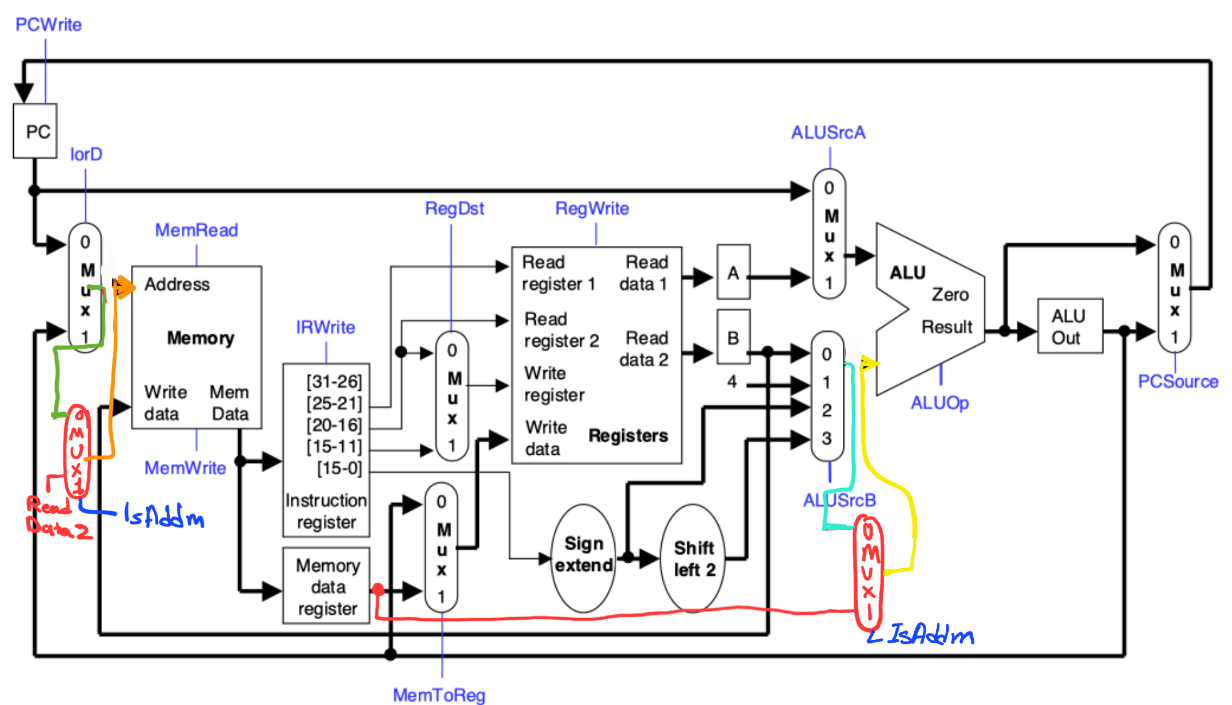
*از ROM دوم برای مشخص کردن شماره FF بعدی استفاده میکنیم. برای ایجاد آن به OpCodeها و FF های فعلی نیاز داریم:*

*ج) طبق صورت سوال تعداد u-instruction هاحداکثر 8 تاست، پس حداکثر 3 بیت برای مشخص کردن آنها نیاز داریم؛ بنابراین برای الف داریم: (نقطه شروع در ROMرا آدرسی میگیریم که 3بیت کم ارزش آن صفر و ۸بیت پرارزش آن Opcodeدستور باشد، و از این آدرس تا ۷آدرس بعدی مربوط به این دستور خاص میشود.)*

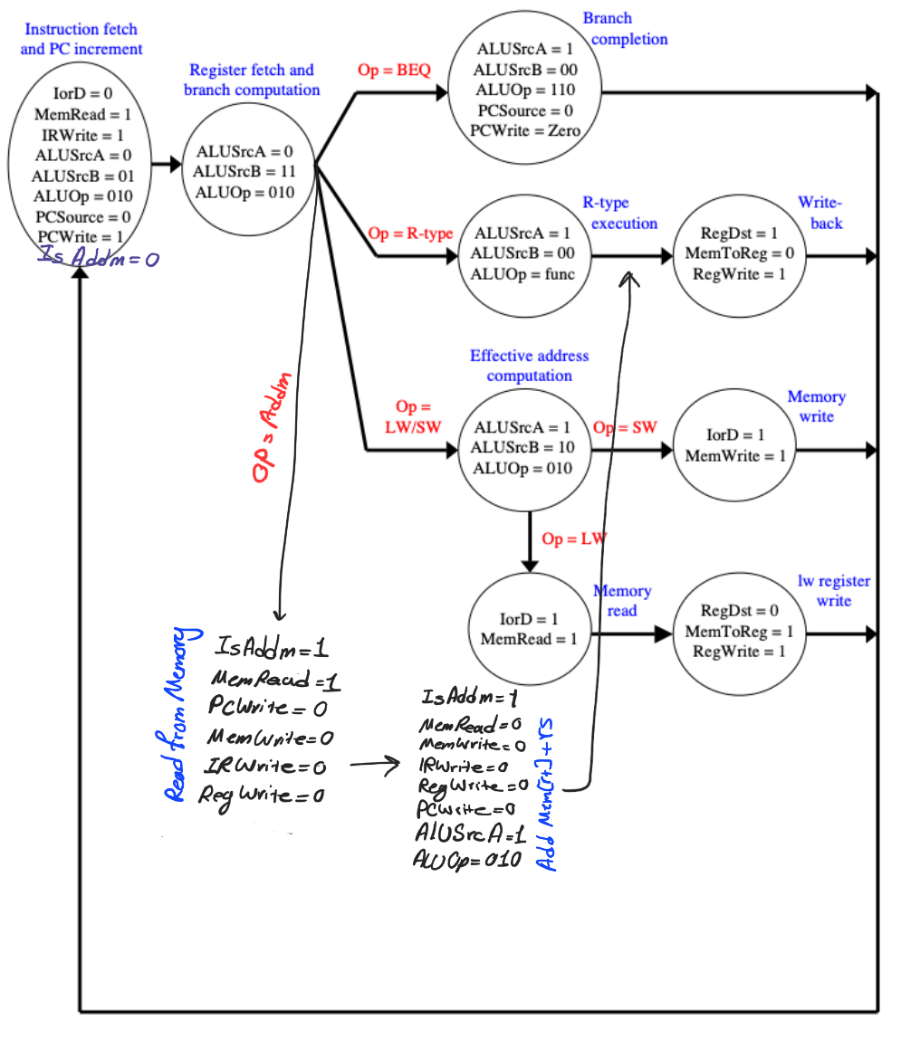
*حال اگر بخواهیم برای ب هم مشابه قبل یک بار سیگنال های کنترلی و بار دیگر state را مشخص میکنیم:*

*و رام دوم:*

1. *آ) مشابه تصویر به 2 تا مالتی پلکسر و یک سیگنال کنترلی جدید نیاز داریم. این سیگنال کنترلی IsAddm در صورت مشاهده OPCode این دستور 1 میشود. این 2 تا مالتی پلکسر نیز یکی برای دادن آدرس درون rt به Memory و دیگری برای دادن Mem[rt] به ALU استفاده میشود که سیمکشای آن را در تصویر میبینید.*



*ب) در تصویر حالت های جدید و سیگنال های مهم در آن حالت را مشخص کردم:*

**