**Computer Architecture Lab**

*Report #1*

*Implementing MIPS*

By: Edris Nasihatkon , Alireza Yazdanpanah

Lab partners

Ahmad Hassani 810194302

Nooshin Taghavi 810194289

پردازنده ی طراحی شده شامل ۵ module با نام های IF ، ID ، EXE ، MEM ، WB است. چهار module اول داری دو module هستند که module اول شامل قسمت های کوچک تری هستند که وظیفه ی خاصی انجام می دهند و module دوم رجیستر بین این قسمت و قسمت بعدی است.

1. IF

* در module ، IFSub یک رجیستر PC داریم که اشاره گر به دستوری است که باید اجرا شود و یک instruction memory که شامل دستوراتی است که اجرا می شود. روند تغییر PC بدین صورت است که بعد از load شدن هر دستور چهار تا اضافه می شود و به دستور بعدی اشاره می کند، مگر آنکه دستور branch باشد. برای اینگونه دستورات یک MUX در ورودی رجیستر PC وجود دارد که برنامه به درستی اجرا شود.
* در module ، IFReg رجیستر های این قسمت و module بعدی است. که شامل PC و Instruction است.

1. ID

* در module ، IDSub شاهد چند قسمت هستیم :
  + Controller که سیگنال های کنترلی را با توجه به opcode دستور مربوطه تعیین می کند. این سیگنال ها عبارتند از :
    - WB\_EN : آیا باید در register file شود؟
    - MEM\_Signal: مشخص کننده ی دستورات load و store در module حافظه.
    - Branch\_Type: نوع branch را مشخص می کند.
    - EXE\_CMD: دستور اجرا شونده توسط ALU در module ، EXE را مشخص می کند.
    - Is\_imm: آیا دستور immediate است؟
  + RegisterFile، شامل سی و دو رجیستر سی و دو بیتی است. که وظیفه ی خواندن ونوشتن روی این رجیستر ها را بر عهده دارد. مقادیر Reg1 و Reg2 را خوانده و در صورتی که enable آن فعال شود WB\_Data را در WB\_Dest(Dest) می نویسد.
  + signExtend، از عدد شانزده بیتی را با توجه به بیت آخر که بیت علامت است به سی و دو بیت extend می کند.

همچنین دو MUX که بین Reg2 ، خروجی signExtend به عنوان val2 و مشخص کننده ی Dest وجود دارد.

* در module ، IDReg سیگنال های کنترلی و Dest، PC، Val1، Val2، Reg2 را در خود رجیستر می کند.

1. EXE

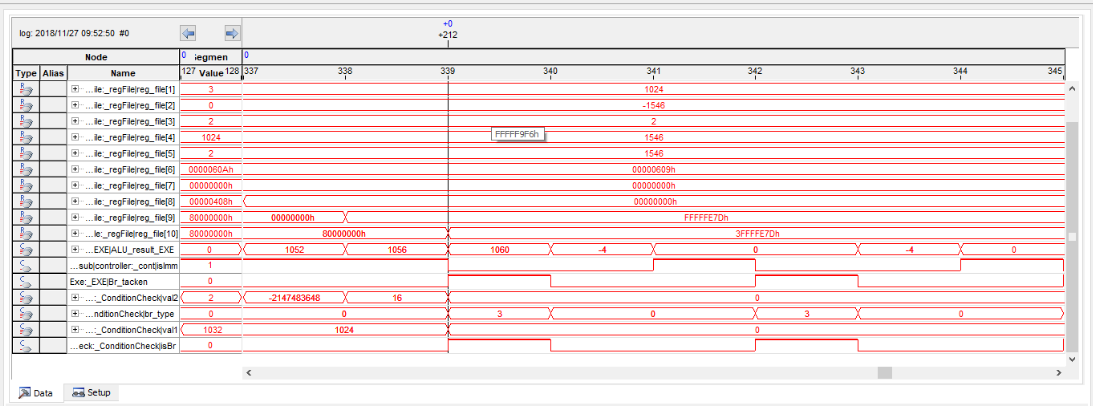
* در module ، EXESub شاهد چند قسمت هستیم:
  + ALU : وظیفه ی اجرای دستورات محاسباتی روی مقادیر Val1 و Val2 با توجه به سیگنال EXE\_CMD را دارد.
  + AdderBranch : مقدار جدید PC برای branch را مشخص می کند.
  + ConditionCheck : با توجه به نوع branch و چک کردن شرط آن مشخص می کند که باید به دستور خاصی که PC آن توسط AdderBranch تولید می شود، branch کنیم یا خیر
* در module، EXEReg سیگنال های WB\_EN ، MEM\_Signal، Dest، PC، ALU\_Result، Reg2 در خود رجیستر می کند.

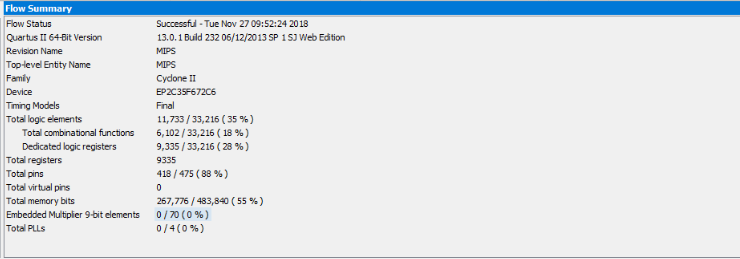
1. MEM

* MEMSub شامل یک حافظه همانند قسمت Register file است که تعداد رجیستر های آن بیشتر است که امکان خواندن و نوشتن روی حافظه را فراهم می کند.
* MEMReg سیگنال های WB\_EN، MEM\_R\_EN، Dest، ALU\_Result ، DataMemOut را در خود رجیستر می کند.

1. WB

شامل یک MUX است که تعیین می کند مقدار ALU\_Result یا DataMemOut در Register file نوشته شود.

◄ Signal Tap

◄ Compilation Report

برای جلوگیری از Hazard بین دستورات لازم سه دستور NOP قرار دادیم که مجموع دستورات به صد و هفده دستور افزایش یافت.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CPI | Runtime | Dedicated logic registers | Total combinatianal functions | Total logic element |
| 342/117 = 2.92 | 342 | 9335 | 6102 | 11733 |

◄ مشکلات

1. اشتباه در متصل کردن سیم ها و فراموشی تعریف wire: بررسی دقیق تر کد و استفاده از modelSim و signalTap
2. در دستوراتی که پرش داشتیم به مشکل برخوردیم که با بررسی دقیق تر متوجه شدیم که با گذاشتن دو NOP مشکل حل میشود و در فاز بعدی متوجه شدیم که میبایست از دستور flush استفاده کنیم.