Subject: گزارش کار پروژه پنجم — مهدی جهانی - علی بهاری

Date: 99.5.16

+ طراحی کش Cache:

با توجه به صورت سوال 4k کش داریم. برای محاسبه‌ی تعداد بلوک‌ها داریم:

$4k \equiv 4 \times 2^{10} = 2^{12}$ ← تعداد بلوک‌ها

به ۱۲ بیت نیاز داریم.

از طرف دیگر برای حافظه، 32k حافظه در اختیار داریم.

برای محاسبه‌ی تعداد بیت tag نیز داریم: $\frac{Mem}{Cache} = \frac{32k}{4k} = 8 = 2^3 \Rightarrow$ 3 بیت هم بلا

3 بیت برای tag نیاز داریم، 12 بیت هم بالا حساب کردیم ← 15 بیت آدرس خواهیم داشت.

از طرف دیگر با داشتن 4k کش، 2^{12} خط برای کش داریم که می‌شود 4096 تا.

به این ترتیب هر بلوک آدرس 15 بیت دارد که 3 بیت آن برای tag استفاده می‌شود.

به عنوان مثال اگر بخواهیم Reset کنیم، تمام ورودی‌های Valid را صفر می‌کنیم.

بیت‌های Valid جزو 15 بیت آدرس نیست و به صورت دستی تنظیم می‌شود.

* نکته: آدرس در نهایت 16 بیت است ← یک بیت برای Valid + ۳ بیت برای tag + ۱۲ بیت آدرس

+ طراحی حافظه اصلی - Main Memory:

Valid را به صورت دستی تعیین می‌کنیم!

با توجه به تمرین قبل 32k حافظه داریم ← $32k = 32 \times 2^{10} = 2^{15}$ ← Memory Size

در هر خط از حافظه اصلی، 32 بیت دیتا داریم. ← 32767 خط داده 32 بیتی خواهیم داشت.

Subject ..... Date .....

در کد تحت Main Memory از ابزار Initial Begin استفاده کرده‌ایم. از آن جایی که آرایه در 8k ذخیره کنیم، از خانه 1024 تا 9125 را در تحت Initial به این کار اختصاص می‌دهیم.

در بخش Initial در هر حلقه از for داده‌ای را در Main Memory می‌نویسیم!

از آن جایی که در هر خط یک دیتا داریم، 4 خط متوالی را می‌خوانیم که یک خانه محسوب می‌شود.

به این ترتیب در خط 16 از main_memory[address+offset] استفاده می‌کنیم و از آن جایی که offset مقدار بین 0 تا 3 دارد، بنابراین 4 خط متوالی را می‌توان خواند!

خط 17 هم برای زمانی است که به کنترلر اطلاع می‌دهد که من آماده‌ام دیتا بنویسی.

* اتصال کش به حافظه اصلی Cache_mainMem_Connection :

در این بخش از کد تنها کش را به حافظه اصلی وصل می‌کنیم. سیگنال‌هایی که از سمت کنترلر می‌آیند، در ابتدای اسم خود C دارند!

در خط 11 برای این که چک کنیم که hit شده یا نه، چک می‌کنیم که tag ای که از کش می‌آید با ۳ بیت آدرس یکی هست یا نیست؟

tag

Subject ..... Date .....

+ اتصال کنترل به طبقه حافظه ← Circuit:

در این قسمت کنترلر را به All Memory وصل می‌کنیم. سیگنال‌های کنترلر بین و داخل حافظه و خود کنترلر توزیع شده‌اند.

به عنوان مثال مشاهده می‌شود که سیگنال‌های کنترلی که در قسمت‌های قبل issue می‌شدند مثل is_wanted_data یا cache_valid وارد کنترلر می‌شوند که از طریق آن‌ها می‌فهمیم hit شده یا خیر؟ یا مثلاً Main_Mem_Ready وارد کنترلر می‌شود که در قسمت‌های قبل آن را دیدیم!

دو سیگنال finish_out, hit_out هم در پایان کنترل مشخص می‌کنند که hit تمام شده یا خیر و دوباره هم مشخص می‌کنند hit انجام شده یا نه؟

+ طراحی قسمت Controller:

- در always اول مشخص می‌کنیم که state فعلی چیست؟
- در always دوم سیگنال‌های کنترلی را issue می‌کنیم.
- در always سوم state بعدی را مشخص می‌کنیم

و در آخر هم یک Counter طراحی کرده‌ایم که در ادامه استفاده‌ی آن را شرح می‌دهیم.

در always اول، ابتدا وضعیت شروع را بررسی می‌کنیم که آیا hit اتفاق افتاده است یا خیر؟

در صورتی که hit شده باشد که به state های بعدی می‌رویم و فرض می‌کنیم کار پایان یافته شده.

در صورتی که miss شده باشد به state بعدی می‌رویم که در خط 13 کلیف آن مشخص می‌شود.

در این جا همان سیگنال main_mem_ready را چک می‌کنیم که آیا می‌توان از حافظه خواند یا خیر؟

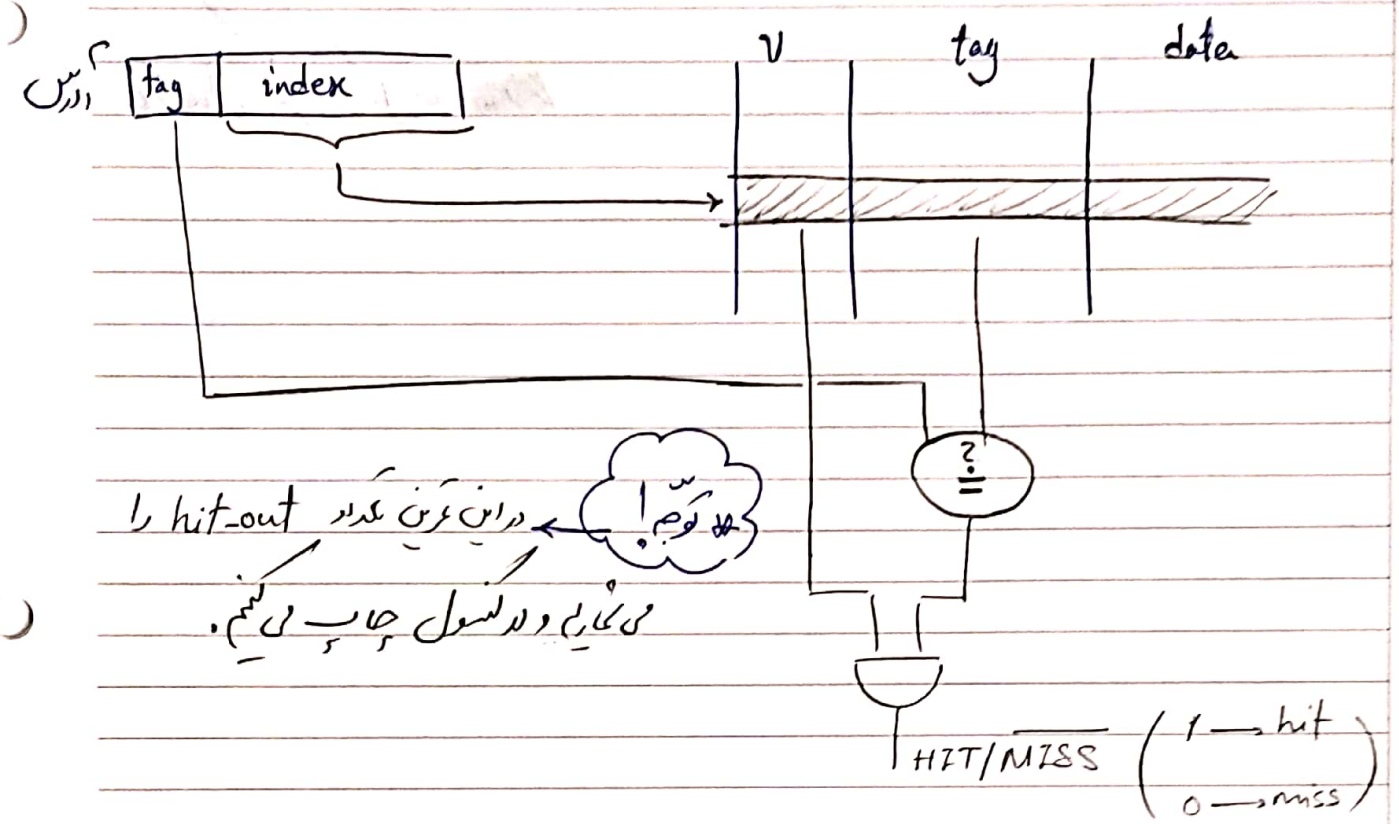
در نهایت چون 4 خط متوالی را باید بخوانیم در خط 15 این کار را انجام می‌دهیم! 4 بار عمل خواندن را انجام می‌دهیم!

← تمت write همان طور always چقدر است که آورده‌ایم تا 4 بار عمل خواندن Counter را انجام بدهد.

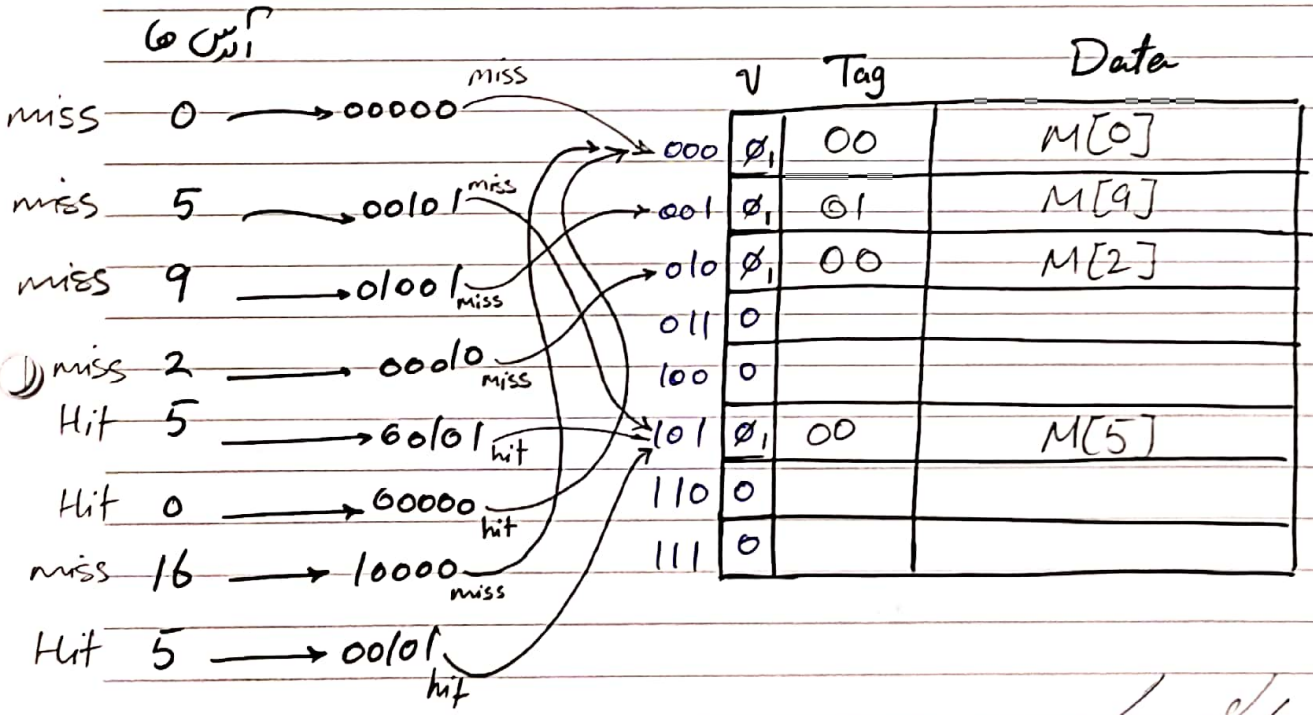
همان طور که پیش از این گفتیم در بلوک always بعدی، سیگنال‌های ترتیبی issue می‌شوند.

مثلاً چک می‌کنیم که hit یا miss اتفاق افتاده است یا خیر؟ ← خط 27

یا در خط 31 وضعیت offset را مشخص می‌کنیم که 4 خط ثبت سرهم را بخوانیم.



* توجه! در این قسمت عدد hit_out را می‌خواهیم و در نتیجه چاپ می‌کنیم.



چون حافظه کش ۸ بلوک Cache داریم ← حافظه اصلی ۳۲ خانه دارد.

$$\text{Hit Rate} = \frac{3}{8} = 0/375 \rightarrow \text{hit rate} = 37,5\%$$

* توجه: طرح بالا مثالی کوچک از پروژه‌ی پیاده‌سازی شده است! (Cache دستی از آن)