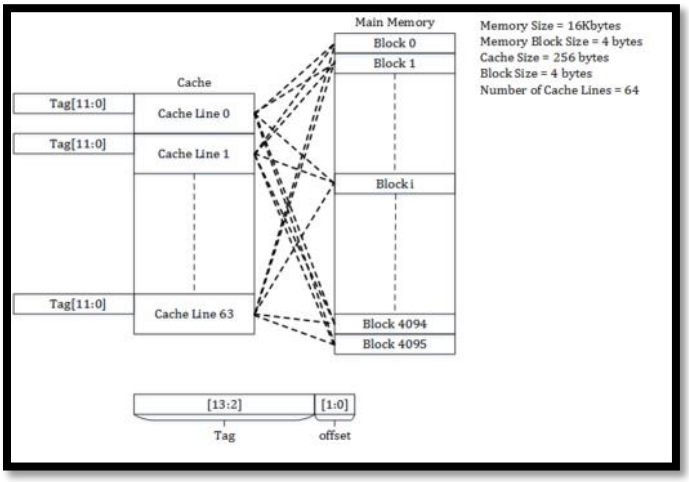
## آزمایش نهم

هدف آزمایش: طراحی حافظه نهان به دو صورت fully associative cache و set associative cache

نحوه انجام آزمایش:

- Fully Associative




برای ساخت این مدل Cache ابتدا چند مدل پورت تعریف میکنیم:

- CLOCK (کلاک)
- Address: یک Vector، ۴ بیتی برای نگه داری آدرس ها
- Read Enable: یک بیت برای تشخیص آنکه داریم آدرس میخوانیم یا خیر
- Data: یک وکتور ۸ بیتی برای نگه داری خود دیتا
- Hit: یک تک بیتی برای تشخیص آنکه Hit داریم یا نه (یادآوری: درحالتی میگوییم Hit داریم که آدرس درخواستی CPU در Cache موجود باشد)
- Write Enable: یک بیت برای تشخیص آنکه آیا داریم در Cache مینویسیم یا خیر
- Data Input: منطقا یک وکتور ۸ بیتی که دیتایی است که میخواهیم نوشته شود

سپس اقدام به طراحی رفتار ماژول میکنیم. برای این بخش ابتدا یک RAM تعریف کردیم. مشخصات این رم به این صورت است:

یک آرایه ۱۶ تایی که هر کدام دارای یک وکتور ۸ بیتی هستند (همانطور که بخش Data در حافظه نهان ۸ بیتی بود)

بنابراین این رم ما ۴ بلوک دارد که هر بلوک ۴ ورد است. به همین دلیل دو بلوک Way0 و Way1 را تعریف میکنیم.

مقدار های Way Validation برای تشخیص معتبر بودن دیتا تعریف میشوند که بعد از این از آنها استفاده خواهیم کرد.

سپس برای هر کدام از Way ها یک Tag ۴ بیتی نیز تعریف میکنیم.

حال به سراغ مرحله اصلی کار یعنی Process میرویم:

درصورتی که کلاک روی لبه بالارونده بود و Read Enable، ۱ بود وارد بدنه شرط میشویم:

متغیر Tag را که پیش از این (خارج از بدنه if) تعریف کرده بودیم برابر با وکتور ۴ بیتی تقسیم هر Address (یادمان باشد که جنس آدرس یک وکتور ۴ بیتی بود) بر سایز رم میکنیم

خب حال ۳ حالت را بررسی میکنیم (با شرط):

- اولین حالت آنکه Tag برابر باشد با Way Zero Tag و همچنین Way اکتیو (Valid) باشد
- دومین حالت آنکه Tag برابر باشد با Way ONE Tag و همچین Way ONE اکتیو باشد
- در غیر این صورت

برای حالت اول و دوم تقریبا یکسان عمل میکنیم با یک تفاوت کوچیک:

ابتدا مود address بر سایز را میگیریم و به Integer تبدیل کرده و در متغیر Modd قرار میدهیم

سپس اندیس Modd در Way مدنظر (وابسته به آنکه در کدام حالت فوقیم) را Fetch میکنیم و در data قرار میدهیم و در آخر Hit را یک میکنیم

و اما حالت سوم

خود این حالت به ۳ حالت تقسیم میشود:

- ابتدا چک میکنیم که Way 0 خالی است یا نه (با استفاده از Valid) اگر بود اندیس ۰ تا ۳ آن را با مقادیر RAM پر میکنیم. برای این پر کردن این عملیات را انجام میدهیم:
  - مقدار Integer تگ را در مقدار Integer سایز ضرب میکنیم (به علاوه یک عدد برای آنکه تعدادی اندیس جلو برویم) که به ما اندیس RAM را میدهم که سپس با خود RAM آن اندیس را Fetch کرده و در Way قرار میدهیم.
  - سپس مقدار Way Valid را یک میکنیم
- حالت دوم همانند حالت اول است اما برای Way 1

- این حالت به این معناست که هر جفت Way ها پر هستند بنابراین باید دیتای جدید RAM را در Cache قرار دهیم که اینکار را با شیفت دادن Way 0 به چپ انجام میدهیم (هر خانه متناظرا وارد Way 1 میشود) سپس کاری که در مراحل ۱ و ۲ انجام دادیم را تکرار کرده (برای Way 0) و تگ جدید را ست میکنیم (برای Way 0)

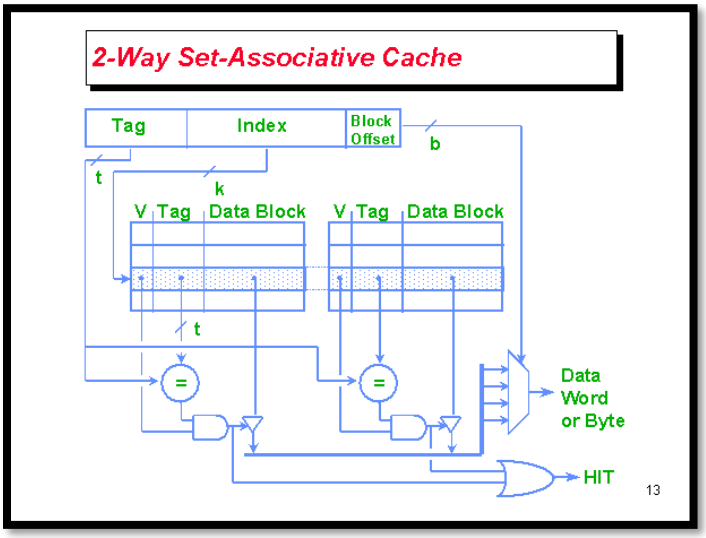
دقت شود که حالت سوم طبیعتا Hit نیست!

برای نوشتن چندان الگوریتم متفاوتی را در پیش نمیگیریم. ابتدا Tag را محاسبه کرده و باز ۳ حالت را بررسی میکنیم. اما اینبار برای حالت ۱ و ۲ علاوه بر بررسی خالی بودن یا خالی نبودن باید چک کنیم که آیا تگ با Way Tag برابر است یا خیر. سپس وارد بدنه شرط میشویم که اینبار بعد از محاسبه Modd مقدار ورودی را در Way(Modd) میریزم و سپس این مقدار را در آدرس مناسبش در Ram قرار میدهیم. Data را نیز معادل همان ست میکنیم و هیت را یک میکنیم.

حال سراغ حالت سوم میرویم: ابتدا چک میکنیم Way خالی است یا نه اگر خالی بود مقادیر رم را مطابق مرحله قبل در آن جایگزاری میکنیم و مطابق حالت ۱ و ۲ همین بخش با استفاده از Modd و دیتای ورودی خود Way و همچین Ram را پر کرده و Valid آن را یک میکنیم. (اینکار را برای Way 0 و Way 1 انجام میدهیم) اما اگر نه Way 0 و نه Way 1 خالی نبودند وارد else آخر میشویم که کارکرد آن ساده است: مانند حالت خواندن یک شیفت میدهیم (FIFO) و مقادیر Way 1 با مقادیر Way 0 جایگزین میشوند. حال با خواندن مقادیر از RAM، Way 0 را که خالی شده بود پر میکنیم و در آخر با استفاده از Modd دیتا ورودی را در اندیس آن پر میکنیم. (هم در RAM و هم در Way). در این حالت Hit صفر است.

- Set Associative

در این مثال خاص یعنی (2 Way)، این دو یعنی Set Associative , Fully هیچ تفاوتی با یکدیگر ندارند بنابراین نیاز به تحلیل جداگانه نیست.



فایل های پروژه در پیوست موجود میباشد

موفق باشید