به نام خدا

فاز دوم پروژه معماری کامپیوتر

نیمسال دوم سال تحصیلی ۱۴۰۰-۱۴۰۱

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

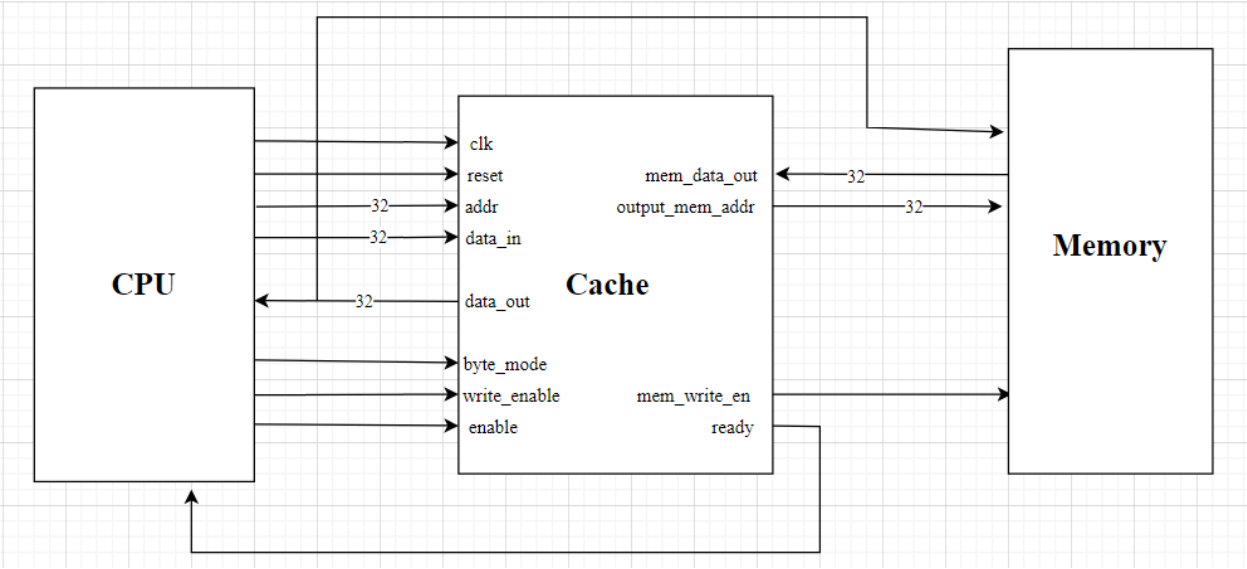
توضیحات فاز:

حافظه اصلی داده شده در فاز قبل در یک کلاک پاسخ را حاضر میکرد. در حالی که در عمل چنین نیست و دسترسی به حافظه اصلی با تاخیر زیادی همراه است. حافظ های که در این فاز در اختیار شما قرار گرفته این تاخیر را شبیه سازی می کند و شما باید از آن به جای حافظه فاز قبل استفاده کنید. این حافظه برای حاضر کردن جواب به چند کلاک زمان نیاز دارد. ابتدا پردازنده خود را به گونه ای تغییر می دهیم که بتواند در دستورات مربوط به حافظه چند کلاک متوقف شود تا عملیات به درستی انجام شود. سپس با پیاده سازی و اضافه کردن یک حافظه ی نهان سطح 1 ، توقفهای به وجود آمده به هنگام دسترسی به حافظه را به حداقل می رسانیم. این حافظه باید از نوع write back باشد اما در مورد ظرفیت و انداز هی بلوک حافظه آزاد هستید و مدل mapping حافظه را هم می توانیدDirect mapped یا set-associative در نظر میگیریم. توجه داشته باشید که حافظه پیاده سازی شده باید در کلاک پیشبینی شده برای پردازنده کار کند. لذا با توجه به این که ابعاد حافظه و اندازهSet ها روی تاخیر آن تاثیر مستقیم دارند، آ نها را با دقت انتخاب کنید. میتوانید حافظه خود را به صورت پارامتری طراحی کنید و با سنتز کردن با مقادیر مختلف مقدار مناسب را بیابید.

اعضای تیم:

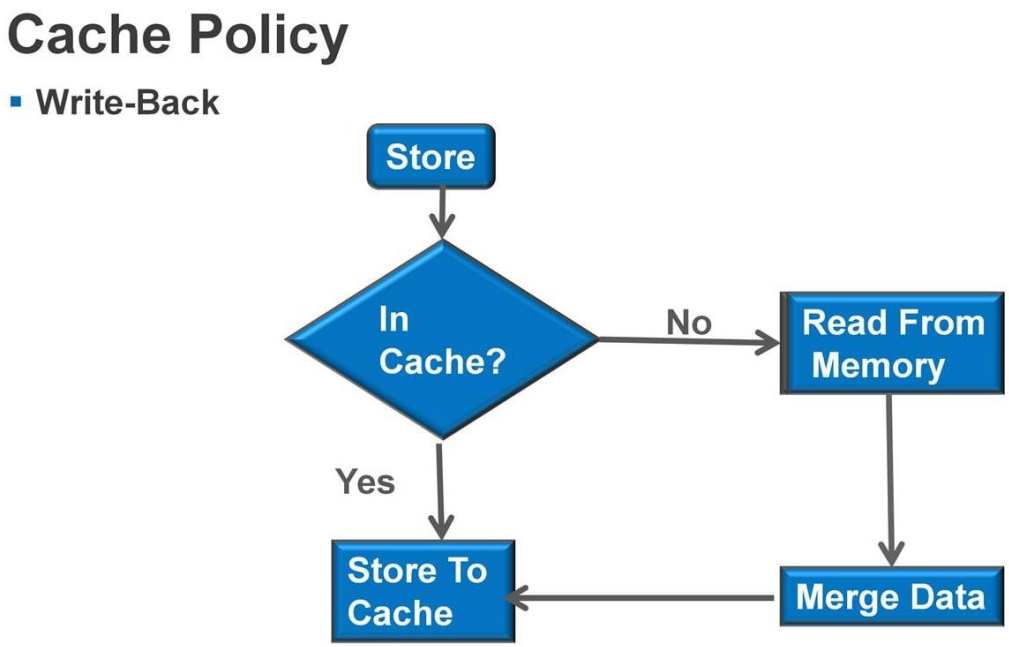
- *سهیل نظری مندجین*
- *بنیامین ملکی*
- *هیربد بهنام*
- *هیراد داوری*

معماری کلی و اتصالات جدید میان memory، cache و CPU به صورت زیر است:



در این معماری cache به عنوان یک واسط اطلاعات مربوط به داده‌های مورد نظر جهت خواندن یا نوشتن در memory را دریافت می‌کند و برای حالت access به داده (read) در صورتی که بلاک آن داده‌ی مورد نظر در cache باشد و tag آن نیز با داده‌ی موجود در cache یکسان باشد، hit رخ می‌دهد و در غیر اینصورت miss رخ داده و لازم است که داده را به cache بیاوریم. به هنگام نوشتن نیز در صورتی که داده در cache موجود باشد (hit رخ دهد)، فلگ dirty آن را فعال می‌کنیم و صرفا در cache مقدار جدید را می‌نویسیم. در حالتی که miss رخ دهد نیز صرفا داده فعلی را یک دور می‌خوانیم و سپس داده جدید را جای آن، در cache، می‌نویسیم. و به طور کلی هنگامی که miss رخ دهد، لازم است که داده‌ی جدیدی در cache آورده شود. اگر ما بلاکی از memory را که قبلا به cache آورده‌ایم تغییر داده باشیم لازم است تا دوباره در memory مقدار ان را update کنیم (به عبارتی صرفا زمانی لامز است که دوباره آن بلاک را در memory بنویسیم که در cache نسبت به زمانی که در ابتدا به cache امده بود، تغییر کرده باشد و این حالتی است که dirty یک باشد). در غیر اینصورت و یا پس از write backکردن، بلاک جدید مد نظر را از memory به cache می‌اوریم.

در حقیقت توضیح بخش آخر بند بالا همان policy write back برای cache است.



در این فاز یکی از مهم ترین مشکلاتی که بهش برخورد کردیم این بود که هنگام نوشتن کش این موضوع که ۴ کلاک طول می کشد تا دیتایی که میخواهیم داخل مموری بنویسیم به رم برسد علاوه بر آن رایت انیبل نیز ۴ کلاک طول می کشد ولی ما ۴ کلاک آن را یک نگه میداشتیم و هنگامی که آن را یک کلاک کردیم این مشکل برطرف شد. برای اینکار در هر کلاک ابتدا write enable را ۰ کرده و هنگامی که میخواهیم write back کنیم یک کلاک آن را یک می کنیم.

```
101                                     state_write_back: begin
102                                             if (clk_counter == 5) begin
103                                                     output_mem_addr = mem_addr; // load the word
104                                                     //$display("starting to load %h", mem_addr);
105                                                     clk_counter = 0;
106                                                     state = state_load;
107                                             end else
108                                                     clk_counter++;
109                                             mem_write_en = clk_counter == 1; // only for one clock!
```

```
50          assign {curr_tag, curr_block, curr_byte_temp} = mem_addr;
51          assign curr_byte = ~curr_byte_temp; // fuck byte ordering
52
53          always @(posedge clk or negedge reset) begin
54                  // always reset these outputs
55                  ready = 0;
56                  mem_write_en = 0;
57                  // check cache reset
58                  if(~reset) begin
59                          for(integer i=0;i<2048;i++) begin
60                                  cache_mem[i] = 0;
61                                  valid[i]=0;
62                                  tag[i] =0;
63                                  dirty[i] =0;
64                          end
```

ساختار و نحوه عملکرد حافظه نهان:

حافظه نهان ما دارای ۲۰۴۸ بلاک بوده که هر بلاک ۱ ورد است.