

Lab 5: Microprocessor Memory Caching 3: The Direct-Mapped & Set-Associative Cache

CME433-01

Addi Amaya

Caa746

11255790

Nov 26th, 2021

Contents

[Single-Line Cache 2](#_Toc86866874)

[Microprocessor Layout 2](#_Toc86866875)

[My Program Sequencer 2](#_Toc86866876)

[Questions 4](#_Toc86866877)

[Question 1 4](#_Toc86866878)

[Question 2 4](#_Toc86866879)

[Question 3 4](#_Toc86866880)

[Question 4 4](#_Toc86866881)

[Cache Memory Benefits 6](#_Toc86866882)

[My Program Sequencer 6](#_Toc86866883)

[Questions 11](#_Toc86866884)

[Question 1 11](#_Toc86866885)

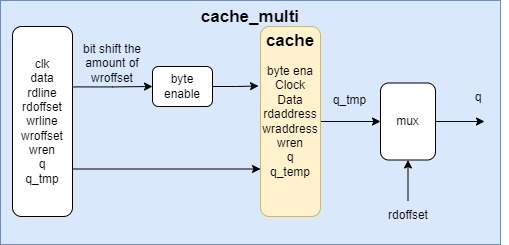
[Question 2 11](#_Toc86866886)

[Question 3 12](#_Toc86866887)

# Multi-Line Direct-Mapped Cache

## Question 1: Block Diagram

Here you can see that the cache\_multi is wrapper for the cache that takes a series of inputs and encodes them to prepare for the cache. The cache output is a 64-bit output which is truncated depending on the input of rdoffset



## Question 2: Cache Operation

Direct-mapped cache refers to a piece of memory that has sections and each section is tied to a part of the main memory. The cache takes in address and takes the 3LSB and creates an index and with the 2MSB creates the tag. The index is checked to the address in the cache, if the tag is the same it is hit otherwise it’s a miss. Hit referring to that an address already exists in the cache and miss saying the opposite.

Example:

The photos below demonstrate a simple example of how direct-mapping works. Address 10011 is broken into an index and tag. The index would be 011 and the tag is 10. So it checks the index of the initial (row 4) then it checks the tag. Because the initial tag is 11 and the given address tag 10, then it’s a miss because they aren’t the same. The opposite happens for address 00001. it has an index of 001 and a tag of 00. So it checks index 001 (row 2) then it checks the tag but because the tags are the same it’s a hit and thus ignored

Shape

Description automatically generatedTable

Description automatically generated

## Question 3: Comparison

### Multi\_cache Flow summary

Total Logic elements: 315

Total Registers: 76

Total Pins: 145

Memory bits: 2368

### Single\_cache

Total Logic elements: 428

Total Registers: 66

Total Pins: 138

Memory bits: 2624

### Observations

You can see above that the single cache has more logic elements, registers and memory bits but it has fewer total pins used. The memory bits for the multi-cache are less meaning it is requiring less memory to function.

## Question 4: End of Execution

### Multi-cache

Graphical user interface

Description automatically generated

### Single-cache

A screenshot of a computer

Description automatically generated with medium confidence

### Observations

There are a few things to note. One is being how long the delays are with respects to eachother. The single cache has a longer delay. Another thing to note is that the pm\_data and ir are the same values the only difference, again, is the delay they have on them. One difference between the two is the in the multi-cache there is an oscillation around 0xa1 for the rom address whereas for the single there is a constant 0xa2. Another observation is that even though the delay in the multi-cache is short is it still keeping up with the writing and reading