Computer Architecture, Section 379: Homework #2

Yousef Alaa Awad

September 28, 2025

1

Given: Assume we have a 64KB 8-way associative cache with 32B blocks

A) What is the set number we are accessing for virtual address 0x08B56A?

B) Wgat is the set number and the tag if the for virtual access in part a if cache is 2-way associative?

C) Repeat Part B for direct mapped cache.

D) What is the effect – between Part A to Part C – (i.e. when reducing associativity) on the following metrics: miss rate, and hit time.

2

Given: Assume page table and a 64-bit processor with 4KB pages. The page table root address is at address 0xFF2400, and each page tabe entry (PTE) is 8 bytes.

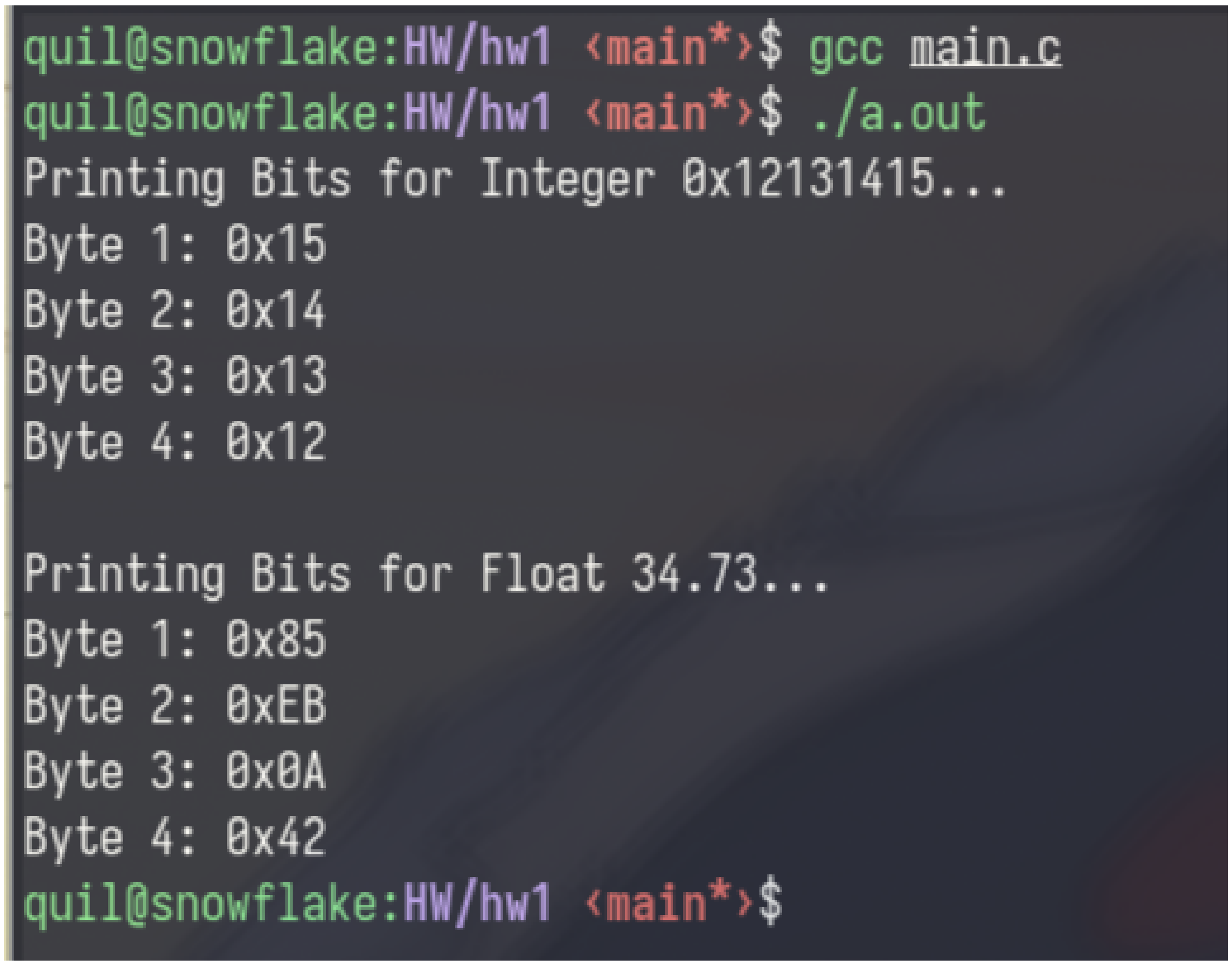
A) What is the size of the hypothetical linear page table?

B) If four-levels page table is used, what are the indices of the corresponding PGD, PUD, PMD, and PTE when accessing 00x0542354C1508?

3

Given: Consider a memory system with the following paramaters:

- Translation Lookaside Buffer has 512 entries and is 2-way set associative.
- 64KByte L1 Data Cache has 128 byte lines and is also 2-way set associative.
- Virtual addresses are 64 bits and physical addresses are 32 bits.
- 4KB page size.



```
quil@snowflake:HW/hw1 ‹main*›$ gcc main.c
quil@snowflake:HW/hw1 ‹main*›$ ./a.out
Printing Bits for Integer 0x12131415...
Byte 1: 0x15
Byte 2: 0x14
Byte 3: 0x13
Byte 4: 0x12

Printing Bits for Float 34.73...
Byte 1: 0x85
Byte 2: 0xEB
Byte 3: 0x0A
Byte 4: 0x42
quil@snowflake:HW/hw1 ‹main*›$
```

Listing 1: My C Code

```c
#include "stdio.h"
#include "stdint.h"

int main()
{
  // Variables
  int32_t sampleInt = 0x12131415;
  float sampleFloat = 34.73;
  // Pointer for individual finding of 2 bytes
  uint8_t *bytePointer = (uint8_t *)&sampleInt; // typecasting for funny reasons

  // Printing out the thing we are wanting
  if (printf("Printing Bits for Integer 0x12131415...\n") != 40)
  {
    return 1;
  }
  // Individually printing out 2 bytes at a time for Integer
  for (int i = 0; i < 4; i++)
  {
    if (printf("Byte %d: 0x%.2X\n", i+1, bytePointer[i]) != 13)
    {
      return 1;
    }
  }

  // Printing out the thing we are wanting x2
  if (printf("\nPrinting Bits for Float 34.73...\n") != 34)
  {
    return 1;
  }
  // Moving pointer to the float
  bytePointer = (uint8_t *)&sampleFloat;
  // Individually printing out 2 bytes at a time for Floats
  for (int i = 0; i < 4; i++)
  {
    if (printf("Byte %d: 0x%.2X \n", i+1, bytePointer[i]) != 14)
    {
      return 1;
    }
  }
  // Return success code
  return 0;
}
```