Homework #4

Rebah Özkoç 29207

Assigned: 25/05/2023
Due: 08/06/2023

1. (**15 pts**) Consider a processor with the following parameters:

Base CPI, no stalls due to cache misses	0.5
Processor speed	2.5 GHz
Main memory access time	100 ns
First-level cache miss rate per instruction	3%
1st option for the second-level cache	
Direct-mapped: the latency	20 cycles
Local miss rate of the second level cache, direct mapped	50%
2nd option for the second-level cache	
8-way set associative: the latency	50 cycles
Local miss rate of the second level cache, 8-way set associative	40%

a. (**5 pts**) We are considering two alternatives for second-level cache memory as shown above. What are the global miss rates if we use second-level direct-mapped cache (1st option) and second-level 8-way set-associative cache (2nd option)?

Global Miss rate = first level cache miss rate * second level cache local miss rate

Global miss rate if we use 1st option:
First level cache miss rate = 3%
Second level local miss rate = 50%
Global miss rate = 3% * 50% = 1.5%

Global miss rate if we use 2nd option:
First level cache miss rate = 3%
Second level local miss rate = 40%
Global miss rate = 3% * 40% = 1.2%

b. (**10 pts**) Calculate the CPI for the processor in the table using: i) only first-level cache, ii) a second-level direct mapped cache, and iii) a second-level 8-way set associative cache.

Processor speed = 2.5 GHz => processor period =1/frequency = 1/(2.5 * 10^9) s = 0.4 ns
Main memory access cycle count = 100 / 0.4 = 250 cycles

i) Only first-level cache:
CPI = Base CPI + miss penalty per instruction
CPI = 0.5 + 3% * 250 = 8

ii) Second-level direct mapped cache:
CPI = Base CPI + miss penalty per instruction
CPI = 0.5 + %3 * 20 + 1.5% * 250 = 4.85

iii) Second-level 8-way set associative cache:
CPI = Base CPI + miss penalty per instruction
CPI = 0.5 + 3% * 50 + 1.2% * 250 = 5

2. (**15 pts**) Assume a direct-mapped cache with 16-byte cache lines. The following code is written in C programming language, where elements of integer arrays of **A** and **B** within the same row are stored contiguously in memory.

```
for(i=0; i<8; i++)
    for(j=0; j<8; j++)
        A[i][j] = A[i][j] + B[i]
```

Assuming there is no conflict and capacity misses, compute the miss rates for this code due to compulsory cache misses.

16-byte cache lines can hold 16/4 = 4 words (integer) per line. For every cache miss we will load 4 words to the cache.

We need to calculate the miss rates for arrays A and B separately.

For A:
A[i][j] will start from A[0][0] and go to A[7][7] => total 64 integer accesses
First access will be a miss then the consecutive 3 accesses will be a hit.
For all 4kth access 4kth access will be a miss and then 4k+1, 4k+1, 4k+3 will be a hit.
We will have 64/4 = 16 misses in array A.

For B:
B[i] will start form B[0] and go to B[7] => total 8 integer accesses
B[0] will be miss and B[1], B[2], B[3] will be a hit.
B[4] miss. B[5], B[6], B[7] hit.
After the first accesses each element will be in cache and there won't be any cache miss.
In B we will have 64 accesses and 2 of them will be a miss.

In total there will be 64 + 64 = 128 accesses and 16 + 2 = 18 compulsory cache misses.
Miss rate = 18/128

3. (**12 pts**) Consider Intrinsity FastMath Processor that implements MIPS 32 instruction set architecture. Its virtual addresses are 32-bit integers. It uses 16 KB pages. The processor has a cache with 256 entries where the block size is 16 words (64 B). Assume that a virtual address is 32-bit number, shown as Address[31:0]. Adopting the notation used for byte offset, Address[1:0] for instance, answer the following questions.

a. (**3 pts**) Give the address bits, namely Address[?:?], for cache index.

Cache block size = 16 words = 64 bytes
Block offset => log2(64) = 6 bits
Two last bytes are byte offset
256 cache lines => log2(256) = 8 bits

Byte offset: [1:0]
Block offset: [5:2]
Cache index: [13:6]

b. (**3 pts**) Give the address bits, namely Address[?:?], for page offset.

Page size = 16 KB = 2^14 B
To represent 2^14 bytes we need 14 bits.

Page offset = [13:0]

c. (**3 pts**) Give the address bits, namely Address[?:?], for virtual page number.

32-bit virtual memory addresses
14 bits page offset
Virtual page number representation = 32-14 = 18 bits

Virtual Page Number = [31:14]

d. (**3 pts**) Give the address bits, namely Address[?:?], for block offset.

16 words per block
4 bytes for representation of each word

Byte offset: [1:0]
Block offset: [5:2]

4. (**15 pts**) Consider a hard disk with the following parameters:

- Rotation speed: 7200 RPM
- Average seek time: 5.8 ms
- Transfer rate: 2 MB/s
- Controller overhead: 8 ms
- Sector size: 4096 B

a. **(7 pts)**What is the average time to read or write a sector from a disk?

Average Read/Write Time of a sector from a disk = Queuing Delay + Controller Time + Seek time + Rotational Latency + Transfer time
Queueing Delay = 0 ms
Controller Time = 8 ms
Seek Time = 5.8 ms
Rotational Latency = (60000*0.5) / 7200 = 4.16666667 ms
Transfer time = 4096 B / 2 MB/s = 2.04 ms
Total time = 0 + 8 + 5.8 + 4.16 + 2.04 ms = 20 ms

b. **(8 pts)** If the transfer rate increases to 4 MB/sec and the control overhead decreases to 6 ms, how much faster is the new disk system?

New read/write time of a sector:
Controller Time = 6 ms
Seek Time = 5.8 ms
Rotational Latency = 4.16 ms
Transfer Time = 4096 B / 4 MB/s = 1.02 ms
Total time = 6 + 5.8 + 4.16 + 1.02 ms = 16.98 ms

5. Answer the following questions (**15 pts**)

a. (**5 pts**) Assume that you have a hard disk with MTTF = 400,000 hours/failure. Calculate the annual failure rate of the disk (AFR).

1 year to hours = 8765 hours
Failed disks per 1000 disks = (1000 drives * 8765 hours/drive) / (400,000 hours/failure) = 21.9125 failures
AFR = 2.1%

b. (**5 pts**) You are asked to construct a **RAID1** disk system using two disks with capacities 100 GB and 150 GB, respectively. Find the total usable size.

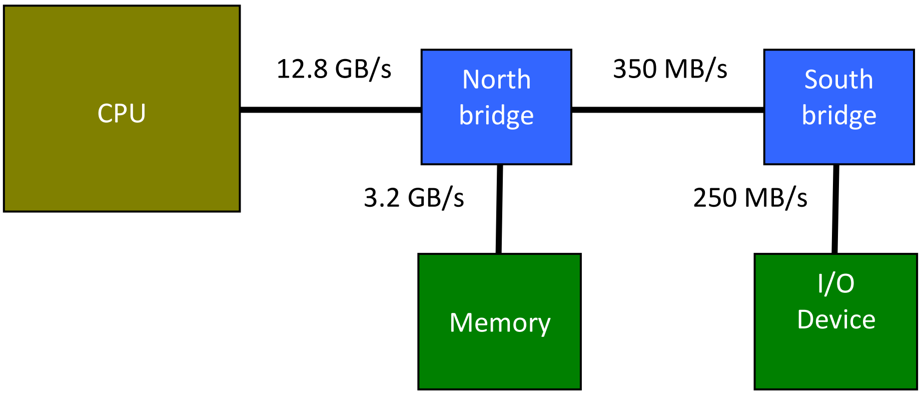
RAID1 system keeps two copies of each piece of information on two separate disks. If there were two disks with capacities 100 GB and 150 GB, we would have 100 GB total usable size.

c. (**5 pts**) Disk failures are not independent and if one disk fails the other fails with 50% probability, as well. What is the overall failure rate of **RAID1** you constructed in (b) using the AFR you find in (a)?

AFR for a disk = 2.1%
To have failure in the whole system both disks must fail.
There are two cases:
The first disk fails (probability 0.021), and then the second disk fails given the first has failed (probability 0.5).
The second disk fails first (probability 0.021), and then the first disk fails given the second has failed (probability 0.5).

Probability of failure of RAID1 = 0.021 * 0.5 + 0.021 * 0.5 = 0.0105 + 0.0105 = 0.021 = 2.1%.

6. (**15 pts**) Consider the computer system shown in the figure below:



The bandwidths of the devices are:

Device	Bandwidth (Byte/s)
Front Side Bus	12.8×10^9
Memory	3.2×10^9
North-South Bus	350×10^6
I/O Device	250×10^6

Assume that the CPU needs an average of 4 bits from memory and 0.2 bits from the I/O device in order to execute an instruction while running a program.

a. (**10 pts**) Show the maximum instruction execution rates (in MIPS) that each device can sustain. What is the maximum sustainable MIPS of the system (assume that the CPU is not the bottleneck in the system)?

Front Side Bus:
Front Side Bus bandwidth = 12.8*10^9 Byte/s = 12.8*10^9 * 8 bit/s
The CPU requires 4 bits from memory and 0.2 bits from I/O device. To maximize the MIPS, we should minimize communication.

The maximum instruction execution rate that front side bus can sustain = (12.8*10^9 * 8) / 0.2 = 512 * 10^9 instruction/s
Converting to MIPS: (512 * 10^9) / 10^6 = 512000 MIPS

Memory:
Memory bandwidth = 3.2 * 10^9 Byte/s = 3.2 * 10^9 * 8 bit/s
The CPU requires 4 bits from memory to execute one instruction.

The maximum number of instructions executable per second = (3.2 * 10^9 * 8) / 4 = 6.4 * 10^9 instruction/s
Converting to MIPS: (6.4 * 10^9) / 10^6 = 6400 MIPS.

North/South Bridge Bus:

North/South Bridge Bus bandwidth = 350*10^6 Byte/s = 350*10^6 * 8 bit/s
The CPU requires 0.2 bits from I/O devices to execute one instruction.

The maximum number of instructions could be executable per second = (350*10^6 * 8) / 0.2 = 14 * 10^9 instruction/s
Converting to MIPS: 14 * 10^9 / 10^6 = 14000 MIPS

I/O device:
I/O device bandwidth = 250*10^6 Byte/s = 250*10^6 * 8 bit/s
The CPU requires 0.2 bits from I/O device to execute one instruction.

The maximum number of instructions executable per second = (250*10^6 * 8) / 0.2 = 1 * 10^10 instructions/s
Converting to MIPS: 1 * 10^10 / 10^6 = 10000 MIPS

The maximum sustainable MIPS of the system is bounded by the memory, and it is 6400 MIPS.

b. (**5 pts**) Suppose that the CPU has a clock speed of 1.8 GHz and it can execute 3 instructions per cycle on average (since it is a multi issue processor). What is the bottleneck in the system now?

1.8 Ghz = 1.8 * 10^9 cycle/s => 3* 1.8 * 10^9 instruction/s
In MIPS => 3* 1.8 * 10^9 / 10^6 = 5400 MIPS

Now the bottleneck of the system is CPU.

7. (**15 pts**) Assume that a program executes in 100 seconds, where 80 seconds is CPU time and the rest of the time is spent for I/O operations. If the CPU performance improves by a speedup value of 1.5 and I/O performance improves by a speedup value of 1.1 per year, how much faster will the program run after 5 years?

After n years	CPU Time (s)	I/O Time (s)	Elapsed Time (s)
0	80	20	100
1	53.33	18.18	71.51
2	35.55	16.52	52.07
3	23.7	15.02	38.72
4	15.8	13.65	29.45
5	10.53	12.41	22.94

Speedup = Old execution time/ New execution time

For CPU:
1.5 = 80/X => X = 53.33 seconds
1.5 = 53.33/X => X = 35.55 seconds
1.5 = 35.55/X => X = 23.7 seconds
1.5 = 23.7/X => X = 15.8 seconds
1.5 = 15.8/X => X = 10.53 seconds

For I/O:
1.1 = 20/X => X = 18.18 seconds
1.1 = 18.18/X => X = 16.52 seconds
1.1 = 16.52/X => X = 15.02 seconds
1.1 = 15.02/X => X = 13.65 seconds
1.1 = 13.65/X => X = 12.41 seconds

Speedup = 100 seconds / 22.94 seconds = 4.36 times