UMass Boston CS 444
Homework 2
Posted Tuesday, February 25, 2025
Due Wednesday, March 5, 2025 at 11:59 pm

Homework must be typed and converted to Portable Document Format (PDF), see `https://en.wikipedia.org/wiki/PDF`. If you have a problem, request an extension before the work is due and explain how much you have done as well as your reason. For hw2, there will be fewer extensions granted. We use the Linux servers of the Computer Science Department to collect homework.

To submit your homework, prepare one PDF file called `hw2.pdf` — the filename must be exactly `hw2.pdf`, otherwise it will not be collected. Upload the file to the `cs444` folder linked to your home directory on the CS Linux server. If you have trouble with uploading, email `operator@cs.umb.edu` for help.

The questions in this homework are based on the Hamming slides, the reading in Tanenbaum and Bos for Chapter 3 and the Ch3 slides.

1 Hamming Code

1.1

For Hamming(15,11), what is the code for binary 1010 1001 010? What is the codefor binary 1010 1101 010?

1.2

What is the Hamming distance between the codes above?

2 Paged Virtual Memory

A computer has a paged virtual memory of 2^{48} bytes – so a virtual address has 48 bits. The page size is 8KB. There is 64GB of RAM. The allocation of a virtual page to a page frame is by direct mapping. Specifically, the lowest X bits of the virtual address are the offset within the page, the next Y bits are the page frame number, and the highest (48 - X - Y) bits are unused in the mapping.

2.1

What is the value of X?

2.2

What is the value of Y?

2.3

A process generates the virtual address 0x7E9C3A5F4B6D – this is in hexadecimal or base 16. What is the offset in hex? What is the page frame number in hex?

3 Page Referencing and Page Faults

Assume there are four page frames. They are empty at the beginning. Consider the following page reference sequence: {2, 1, 3, 4, 2, 1, 3, 2, 6, 1, 3, 2, 1, 5, 3, 2, 1, 4}.
If we use Least Recently Used (LRU) page replacement algorithm, how many page faults occur?

4 Small Example of Virtual Address Space

Instead of a 64-bit machine, consider a 4-bit machine. This means addresses anywhere can only have 4 bits. The largest address is 1111 in binary or F in hex. The total number of addresses is 16, from 0000 to 1111.
Further, consider bytes of memory with addresses. This means the machine is byte addressible. Therefore, the number of bits in the address limits the number of bytes in memory.
In the example, we show address in hex and in binary. We show data in binary. There is data in only the low 8 bytes to start so the other bytes are blank. Here is the example:

```
Address:        Data:
Hex:  Binary: Binary:
F     1111      not set
E     1110      not set
D     1101      not set
C     1100      not set
B     1011      not set
A     1010      not set
9     1001      not set
8     1000      not set
7     0111      11001100
6     0110      00110011
5     0101      11110000
4     0100      00001111
3     0011      10101010
2     0010      01010101
1     0001      11111111
0     0000      00000000
```

4.1

Show the example data moved to start at address 6 hex. What is the address of the byte that is highest in memory now?

4.2

Login to the server and display a text file using the command `hexdump -C <filename>.txt`. Type two single complete lines of exactly what you see, from the left to the right, including address in hex, data in hex and data in ASCII:

Identify the format used for each group, meaning state specifically something like: `On the left, the first n characters is ... The middle group includes... The group on the right is ....`