The second exam is Thursday, March 10. It will cover topics from Homework 2 through Homework 4, including anything pertaining to binary representation. Anything covered in lecture on the topics covered in Homework 2 - Homework 4 is fair game.

Main Test Topics
- hexadecimal number system
- binary-arithmetic overflow
- cache access time questions
- Graphics (how graphics are stored in the computer)
- Networks:
  - definitions
  - modem speed questions
- Web/Internet

Hexadecimal:
Hexadecimal is base 16
There are 3 types of problems involving hexadecimal (hex) that you should be able to do:
- convert decimal to hexadecimal
- convert hex to decimal
- convert binary to hex

Converting Hexadecimal to Decimal:
First we learned how to convert hexadecimal to decimal. Recall that each position in a hexadecimal number represents a multiple of a power of 16:

\[
\begin{array}{cccc}
256 & 16 & 1 & \\
16^2 & 16^1 & 16^0 & \\
\end{array}
\]

Example 1: Convert 100 (hexadecimal) to decimal:

\[
\begin{array}{cccc}
1 & 0 & 0 & \\
256 & 16 & 1 & \\
\end{array} = 1 \times 256 + 0 \times 16 + 0 \times 1 = 256
\]

Example 2: Convert 22 (hexadecimal) to decimal:

\[
22 = 2 \times 16 + 2 \times 1 = 32 + 2 = 34
\]

In hexadecimal we need to represent the values 10, 11,..., 15 as single digits so we use letters:

A=10
Example 3: Convert 2C (hex) to decimal:
Recall that C represents 12, so
\[2C = 2 \times 16 + 12 \times 1 = 32 + 12 = 44\]

Converting Decimal to Hexadecimal:
This is trickier. I suggest you do it as follows:
Say you want to convert 50 (decimal) to hexadecimal.

Ask yourself:
1) What is the largest power of 16 that goes into this number?
   *For our example of 50, the answer is 16. Draw slots for 1, 16.*

\[ \begin{array}{c|c}
16 & 1 \\\n\end{array} \]

2) Figure out: How many times does this power go into this number?
   *For our example, how many times does 16 go into 50? Answer: 3.*
3) Put that value in the correct position.
   *Example:*

\[ \begin{array}{c|c|c}
3 & 16 & 1 \\
\end{array} \]

 Repeat steps 1-3 with the remainder until the remainder is 0.
*In our example the remainder of 50/16 is 2 since 3*16 = 48. Since 2 < 16, we can put “2” in the 1’s position giving: 32 as our answer. 50\text{\_}_{10} = 32\text{\_}_{16}*

Example 1: Convert 66\text{\_}_{10} to hexadecimal (the 10 means that this number is base 10).
The largest power of 16 that goes into 66 is again 16 (256 doesn’t go).
How many times does 16 go into 66? Answer: 4. 4*16 = 64.
So we put a 4 in the 16 position and the remainder 2 in the 1’s position:
\[ \begin{array}{c|c|c}
4 & 16 & 2 \\
\end{array} \]
So 66\text{\_}_{10} = 42\text{\_}_{16}

Example 2: Convert 170\text{\_}_{10} to hexadecimal.
16 is the largest power of 16 that goes into 170. It goes 10 times. We use A to represent 10, so we put an A in the 16 position. What is the remainder? 170 = 10*16 + 10 so the remainder is 10 and the answer is AA.
\[ \begin{array}{c|c}
A & A \\
\end{array} \]
16 1
Check the answer to see if it is correct: 170=10*16+10
Convert 8 bit binary to Hexadecimal:
Given an 8-bit binary string, what is the equivalent value in hexadecimal? You could calculate this by converting the binary to decimal and then converting the decimal to hexadecimal. There is an easier way - a trick.

How (roughly):
1) Divide the bits into groups of 4 bits.
2) Calculate the decimal equivalent of each group of 4 bits. Put the hexadecimal character for that decimal number in the position under the 4 bits.

Example 1: Convert 11100110 to hexadecimal.
Given: 11100110
Divide into 2 groups: 1101 0110
The first group, 1101 = 8 + 4 + 1 = 13, which is D in hexadecimal.
The second group 0110 = 4 + 2 = 6 in hexadecimal.
So 11010110 = D6 in hexadecimal.

Example 2: Write 10011111 as hexadecimal.
10011111 = 1001 1111; 1001 = 9 1111 = 15, which is represented in hex as F.
So the answer is 9F.

Example 3: Write 00001111 in hexadecimal.
First divide into 2 groups: 0000 1111.
We know 0000 = 0 and 1111 = F so the answer is 0F.

Arithmetic overflow
When you are doing math using almost any kind of machine (like a calculator or a computer) there is a limited amount of storage for any number. Where I could continue writing 0's after a number to make it longer in real life, a computer just has a certain number of bits to hold any number. Imagine a calculator that only had 3 spaces to hold decimal digits. If you tried to add 900 and 100, both of which just take 3 spaces, then you would get 1000, which takes 4 spaces. If there was only room for 3 spaces, then you would likely lose the 1 in 1000, giving a wrong answer.

So arithmetic overflow occurs when a program tries to store a value in a memory cell that is too large for the memory cell. This can lead to serious numeric errors.

For example, if you have 8 bits to store a positive unsigned integer, then the largest value you can store is: 11111111 which is 255 in decimal. Any number 255 or smaller can be stored successfully in 8 bits. But what if you try to add two of these numbers? In some cases will you have a problem? What if you try to add 100 and 200? You can represent both of these numbers in binary in 8 bits, but when you add them you get 300 which is too big for 8 bits and you would lose any bits above the 8th bit and get a wrong answer.

Possible Questions:
If you have 8 bits which of these additions will produce an overflow?
10+20
100+100
200+300
200+60
100+100

Answer:
200+300 and 200+60

**Cache Access Time**
Recall what cache is and how we use it. It is a small amount of fast memory used to help speed up accesses to memory. When we only had RAM, it was easy to say how long it took to access memory - it was whatever the speed of RAM was. When we are using cache, it is more complicated. Assume you are using a cache and want to access something in memory. Instead of going straight to RAM to get it, you first check cache to see if it is there. If so, good - return the value. If not, go to RAM to get the value.

We can't calculate the exact time to access data now, because it depends on an unknown - whether the data is in cache or not. But we can compute the average (or expected) cache access time if we know the following data: the cache access time, the RAM access time, and the hit rate. The hit rate is the percentage of time that the data that we are looking for is in cache. You can express it as a percent, like 70%, but when you calculate with it you must express it as a decimal like 0.7.

A miss is the opposite of a hit - it is when the data we wanted was not in cache. Since every memory access is either a hit or a miss, we have:

\[ \text{miss rate} = 1 - \text{hit rate} \]

How should we calculate the average cache access time? We use the following formula:

\[ \text{Avg access time} = (\text{time for a hit}) \times \text{hit rate} + (\text{time for a miss}) \times \text{miss rate} \]

The time for a hit = the time to access cache, which will usually be given to you in the question. What is the time for a miss? Since we have to check cache first to see if the data is in there, the time for a miss is (cache access time + RAM access time).

**So the formula is:**

\[ \text{Average access time} = (\text{time}_{cache}) \times \text{hit rate} + (\text{time}_{cache} + \text{time}_{RAM}) \times \text{miss rate} \]

and you just plug into this formula.

**Example:** Homework 2, Problem 5:

RAM access time=40ns Cache access time =10ns
Hit rate=60%

Plugging into the formula:

Avg. Access Time = 10 * 0.6 + (10 + 40) * 0.4 = 6 + 20 = 26 ns.
In these problems we will either give you the values (cache time, RAM time, hit rate) or could make you calculate one from another. What if we said the cache access time is 10 ns and RAM is 3 times slower than cache. Then you should be able to figure out that the RAM access time should be $3 \times 10 = 30$ ns.

**Representing Graphics**

2 main formats
- bitmap--pixels
- vector--equations, scalable

Within bitmaps there are different file formats like Jpeg and GIF. Both of these are compression algorithms. A file with a name like `mypicture.jpeg` is likely to have been compressed.

Problems: On the homework, to test your understanding of how graphics can be stored, we asked questions like the following:

**Question:** Suppose you have a 10x10 picture.
If the picture is black and white, where black is represented by a 1 and white by a 0, how much storage does this take (in bytes)?

Answer: Each pixel takes 1 bit (to store a 0 or a 1).
The 10x10 image has 100 pixels.
$100 \text{ pixels} \times 1 \text{ bit per pixel} = 100 \text{ bits}$.
But we wanted the answer in bytes so we have to divide $100 \text{ bits} / 8 = 12.5 \text{ bytes}$. A better answer would be 13 bytes - round up.

**Question:** Same situation, but now we want to store greyscale 0 - 255.
Now each pixel needs 1 byte storage
so a 10x10 picture has 100 pixels & 100 bytes

**Question:** Suppose you still have a 10X10 picture, but now you want to store one of 16 colors with each pixel. If each color has a number between 0 an 15, then you need 4 bits to store the color numbers (0000 to 1111). This is half a byte.
So with 16 colors each pixel needs $\frac{1}{2} \text{ byte}$ storage.
A 10x10 epicure has 100 pixels & therefore, 50 bytes

**Question:** If you wanted to use TrueColor, then you store a red, green, and blue byte with each pixel.
So each pixel needs 3 bytes of storage.
For a 10X10 picture, this would take 300 bytes.

JPEG, GIF are compressed file formats for bitmaps