As you wait for class to start, answer the following question:

- What is important in a computer? What features do you look for when buying one?
Review Problem 1

- Programming languages have many instructions, but they fall under a few basic types. One is arithmetic (+, -, *, /, etc). What are the others?
Review Problem 2

- In assembly, set X0 to \(-X1\).
Review Problem 3

- In assembly, compute the average of positive values $X_0$, $X_1$, $X_2$, $X_3$, and put into $X_{10}$
Review Problem 4

- What would the results of this C++ code be in memory? Assume we start using memory at 1000.

```cpp
struct {char *a, *b;} foo;
foo *obj;
obj = new foo;
obj->a = new char[8];
obj->b = new char[4];
obj->a[1] = 'x';
obj->b[2] = 'y';
```
Review Problem 5

- In assembly, replace the value in X0 with its absolute value.
Review Problem 6

- Register X0 has the address of a 3 integer array. Set X15 to 1 if the array is sorted (smallest to largest), 0 otherwise.
Review Problem 7

- Sometimes it can be useful to have a program loop infinitely. We can do that, regardless of location, by the instruction:

- **LOOP: B LOOP**

- Convert this instruction to machine code

```
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
```
Review Problem 8

- We goofed, and wrote: ADD X0, X1, X4, when we meant to write SUB X0, X1, X4. The instruction is at location Mem[0]. What’s the simplest program to fix this?
Review Problem 9

- What does the number \(100011_2\) represent?
Review Problem 10

- Perform the following binary computations.

\[
\begin{array}{c}
1 & 0 & 1 & 1 & 0 \\
+ & 0 & 0 & 1 & 1 & 1 \\
\hline
& 1 & 0 & 0 & 1 & 1
\end{array}
\]
Review Problem 11

- For the buggy majority circuit below, the expected and the measured results are shown in the table. What gate is broken in this circuit?

**Table**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Expected</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Review Problem 12

- How would the ALU be used to help with each of the following branches? The first is filled in for you:
  - B.EQ: SUBS X31, <val1>, <val2>; use zero flag
  - B.NE:
  - B.GE:
  - B.GT:
  - B.LE:
  - B.LT:
Review Problem 13

- Write assembly to compute $X_1 = X_0 \times 5$ without using a multiply or divide instruction.
Review Problem 14

- What aspects of a microprocessor can affect performance?
Review Problem 15

- Orange runs at 1GHz, and provides a unit making all floating point operations take 1 cycle. Grape runs at 1.2 GHz by deleting the unit, meaning floating point operations take 20 cycles. Which machine is better?
Review Problem 16

- If a 200 MHz machine runs \( \frac{1}{2} \) billion instructions in 10 seconds, what is the CPI of the machine?

- If a second machine with the same CPI runs the program in 5 seconds, what is its clock rate?
Review Problem 17

- A program’s execution time is 20% multiply, 50% memory access, 30% other. You can quadruple multiplication speed, or double memory speed
  - How much faster with 4x mult:

  - How much faster with 2x memory:

  - How much faster with both 4x mult & 2x memory:
Review Problem 18

- A RISC machine is shown to increase the instructions in a program by a factor of 2. When is this a good tradeoff?
Review Problem 19

- What is done for these ops during the CPU’s execute steps at right?
  - ADD X0, X1, X2  STUR X3, [X4, #16]  LDUR X5, [X6, #8]
Review Problem 20

- Add the instruction “MUL Rd, Rn, Rm” to the add/sub datapath.
Review Problem 21

- Immediate vals for some instructions are sign-extended, while others are not. Build a 16bit to 64bit sign-extend unit that can handle both.
Review Problem 22

- Develop a single-cycle CPU that can do LDUR and STUR (only). Make it as simple as possible
Review Problem 23

- How would we add the CBGTZ (conditional branch greater than zero) instruction to our CPU?
Review Problem 24

- What mods are needed to support branch register:
  - PC = Reg[Rd]
Show the control settings needed to implement ADDI Rd, Rn, imm12
Review Problem 26

To allow a CPU to spend a cycle waiting, we use a NOP (No operation) function. What are the control settings for the NOP instruction?

<table>
<thead>
<tr>
<th>Reg2Loc</th>
<th>ALUSrc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemToReg</td>
<td>RegWrite</td>
</tr>
<tr>
<td>MemWrite</td>
<td>BrTaken</td>
</tr>
<tr>
<td>UncondBr</td>
<td>ALUOp</td>
</tr>
</tbody>
</table>

![Diagram of control settings for the NOP instruction]
Review Problem 27

- When we discussed inserting registers, we limited it to Acyclic Combinational Logic. Why?
Review Problem 28

- Given what we know about pipelining, assume in a widget factory it takes 40 minutes to make 1 widget. If we pipeline the process into S stages, how long will it take to make N widgets?
Review Problem 29

- The pipelined CPU has the stage delays shown
  - Is it better to speed up the ALU by 10ns, or the Data Memory by 2ns?
- Does your answer change for a single-cycle CPU?
Review Problem 30

- If we built our register file to have two write ports (i.e. can write two registers at once) would this help our pipelined CPU?
Review Problem 31

- What registers are being read and written in the 5th cycle of a pipelined CPU running this code?

```
ADD X1, X2, X3
ORR X4, X5, X6
SUB X7, X8, X9
EOR X10, X11, X12
AND X13, X14, X15
```
Review Problem 32

- ARMv8 has 1-operand branches that test zero/not zero, and conditional branches that use the results of a previous CMP, but no 2-operand branches that compare and branch in one instruction. Why?
Review Problem 33

- Do the unconditional branch instructions (B, BR) have problems with hazards?
Review Problem 34

- What forwarding happens on the following code?

LDUR X0, [X1, #0]
ADD X2, X3, X3
ORR X31, X0, X4
CBNZ X2, END
SUB X5, X31, X2
Review Problem 35

- What should we do to this code to run it on a CPU with delay slots?

```assembly
AND X0, X1, X2
ORRI X0, X0, #7
ADD X3, X4, X5
LDUR X6, [X3, #0]
CBNZ X6, FOO
B BAR
```
Review Problem 36

Why might a compiler do this transformation?

/* Before */
for (j=0; j<2000; j++)
    for (i=0; i<2000; i++)
        x[i][j]+=1;

/* After */
for (i=0; i<2000; i++)
    for (j=0; j<2000; j++)
        x[i][j]+=1;
Review Problem 37

If you can speed up any level’s hit time by a factor of two, which is the best to speed up?

<table>
<thead>
<tr>
<th>Level</th>
<th>Hit Time</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>1 cycle</td>
<td>95%</td>
</tr>
<tr>
<td>L2</td>
<td>10 cycles</td>
<td>90%</td>
</tr>
<tr>
<td>Main Memory</td>
<td>50 cycles</td>
<td>99%</td>
</tr>
<tr>
<td>Disk</td>
<td>50,000 cycles</td>
<td>100%</td>
</tr>
</tbody>
</table>
Review Problem 38

- The length (number of blocks) in a direct mapped cache is always a power of 2. Why?
Review Problem 39

- For the following access pattern, what is the smallest direct mapped cache that will not use the same cache location twice?

  0
  13
  9
  17
  4
  10
  24
Review Problem 40

- A 32-bit CPU has this $2^8$ line DM cache with $2^4$ byte blocks. What memory locations are now held?

<table>
<thead>
<tr>
<th>Valid Bit</th>
<th>Tag</th>
<th>0</th>
<th>1</th>
<th>Data</th>
<th>$2^{4-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$2^8-1$</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Valid Bit: 0 to $2^8-1$
Tag: 20’h00001 to 20’h0BEEF
Data: Empty

Valid Bit: 0 to $2^8-1$
Tag: 20’hFFFFF to 20’hDEAD1
Data: Empty
Review Problem 41

- How many total bits are required for a direct-mapped cache with 64 KB of data and 8-byte blocks, assuming a 32-bit address?

Index bits:

Bits/block:
  Data:
  Valid:
  Tag:

Total size:
Review Problem 42

- In a Fully Associative Cache with 256 lines, and 8-byte blocks, how many bits are the following?
  - Byte Select
  - Cache Index
  - Cache Tag
Review Problem 43

- Assume we have three caches, with four one-word blocks:
  - Direct mapped, 2-way set assoc. (w/LRU), and fully associative
- How many misses will each have on this address pattern:
  - Byte addresses: 0, 32, 0, 24, 32
Review Problem 44

- Which is the best L1 cache for this system?
  - Direct Mapped: 1 cycle, 80% hit rate
  - 2-way Set Associative: 2 cycle, 90% hit rate
  - Fully Associative: 3 cycle, 95% hit rate

<table>
<thead>
<tr>
<th>Level</th>
<th>Hit Time</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>10 cycles</td>
<td>90%</td>
</tr>
<tr>
<td>Main Memory</td>
<td>40 cycles</td>
<td>99%</td>
</tr>
<tr>
<td>Disk</td>
<td>4,000 cycles</td>
<td>100%</td>
</tr>
</tbody>
</table>
Review Problem 45

- Can a direct-mapped cache ever have less cache misses than a fully associative cache of the same capacity? Why/why not?
Review Problem 46

- Assume we have separate instruction and data L1 caches. For each feature, state which cache is most likely to have the given feature
  - Large blocksize
  - Write-back
  - 2-cycle hit time
Review Problem 47

- Here is a graph of runtime vs. \( N \), on a log-log plot, for the following code. Explain

```c
int x[N];
for (j = 0; j < 1000; j++)
    for (int i = 0; i < N; i++)
        x[i]++;```

```c
```
Review Problem 48

- For a dynamic branch predictor, why is the Branch History Table a direct-mapped cache? Why not fully associative or set associative?
Review Problem 49

- How would various branch predictors do on the bold branch in the following code?
  - A 1-bit predictor will be correct ___%
  - A 2-bit predictor will be correct ___%

```c
while (1) {
    if (i<3) counter++;
    i=(i+1)%6; /* i counts 0,1,2,3,4,5,0,1,2... */
}
```
Review Problem 50

- For the constraint graph for this SWAP code, is there an edge between the two SW’s?

1: LDUR X0, [X5, #0]
2: LDUR X1, [X6, #0]
3: STUR X1, [X5, #0]
4: STUR X0, [X6, #0]
Review Problem 51

- Show the constraint graph for this code, indicating the type of hazard for each edge.

1: LDUR X1, [X6, #8]
2: ADD X2, X1, X6
3: LDUR X3, [X7, #16]
4: SUB X4, X3, X8
5: STUR X5, [X9, #0]
6: CBZ X15, FOO
Review Problem 52

Would loop unrolling & register renaming be useful for the following code? If so, what would the resulting code look like?

while (i<400) {
    if (x[i]==CONST) counter++; /* Count number of CONSTs in array */
    i++;
}
Review Problem 53

- In assembly, replace the value in X0 with its absolute value, without using any branches.
Review Problem 54

- A prototype 4-way VLIW has no delay slots, and a CPI of 1.0. What may have caused this?
Review Problem 55

- Intel provided this benchmark. If they are building a superscalar based on this with load/store, branch, and ALU units, what number of each would you suggest?
We added a counter to the multicore code. What will the final value of counter be?

```c
int max(int vals[], int len) {
    int global_result = -infinity;
    int lenT = len/num_procs;
    for (int i=0; i<num_proc; i++)
        process maxT(&vals[i*lenT], lenT);
    int counter = 0;
    while (counter != num_procs)
        wait;
    return global_result;
}

void maxT(int vals[], int len) {
    int my_result = -infinity;
    for (int i=0; i<len; i++) {
        if (vals[i] > result)
            result = vals[i];
    }
    if (my_result > global_result)
        global_result = my_result;
    counter++;
```
Review Problem 57

- Assume you can use any ARM instructions except multiply and divide, but must use only Intels’s 2 operand formats. Compute $X_1 = X_0 \times 5$. If necessary, you can use a MOVE instruction, that copies one register to another.