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, copy the value of register $a0$ to register $t0$
Review Problem 3

- In assembly, compute the average of $a0, a1, a2, a3$, and put into $v0
Review Problem 4

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

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

- In assembly, replace the value in $a0 with its absolute value.
Review Problem 6

- Register $a0 has the address of a 3 integer array. Set $v0 to 1 if the array is sorted (smallest to largest), 0 otherwise.
Review Problem 7

- For the following code, can you ever have labels A1 == A2? A1 == A3? A1 == A4?

```assembly
lw $t0, 100($zero)
beq $t0, $0, A2

A1:
   lbu $t5, 0($t0)

A4:
   beq $t5, $zero, A4
   slt $t6, $t5, $t2
   bne $t6, $zero, A3

A2:
   slt $t6, $t3, $t5

A3:
   add $t5, $t5, $t4
   sb $t5, 0($t0)
   addi $t0, $t0, 1
   j A1
```
Review Problem 8

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

- LOOP: BEQ $7, $7, LOOP

- Convert this instruction to machine code
Review Problem 9

- We goofed, and wrote code with: add $t0, $t1, $t4, when we meant to write sub $t0, $t1, $t4. The instruction is at location Mem[0]. What’s the simplest program to fix this?
Review Problem 10

- What does the number $100011_2$ represent?
Review Problem 11

- Perform the following binary computations.

\[
\begin{align*}
1 & \ 0 & \ 1 & \ 1 & \ 0 \\
+ & \ 0 & \ 0 & \ 1 & \ 1 & \ 1 \\
\end{align*}
\quad
\begin{align*}
1 & \ 0 & \ 0 & \ 1 \\
- & \ 0 & \ 0 & \ 1 & \ 1 \\
\end{align*}
\]
Review Problem 12

- 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>
<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>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</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 13

- How would the ALU be used to help with each of the following branches? The first is filled in for you:
  - beq ($rs == $rt) subtract $rt from $rs, use zero flag
  - bne ($rs != $rt)
  - bgez ($rs ≥ 0)
  - bgtz ($rs > 0)
  - blez ($rs ≤ 0)
  - bltz ($rs < 0)
Review Problem 14

- Write MIPS assembly to compute $t1 = t0 \times 5$ without using a multiply or divide instruction.
Review Problem 15

- What aspects of a microprocessor can affect performance?
Review Problem 16

- 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 17

- 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 18

- 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 19

- 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 20

- What is done for these ops during the CPU’s execute steps at right?
  - add $t0, $t1, $t2  sw $t3, 16[$t4]  lw $t5, 8[$t6]
Review Problem 21

- Add the instruction “mult rd, rs, rt” to the add/sub datapath.
Review Problem 22

- Immediate vals for ADDI are sign-extended, while those for ORI are extended with zeros. Build a sign-extend unit that can handle both.
Review Problem 23

- Develop a single-cycle CPU that can do LW and SW (only). Make it as simple as possible.
Review Problem 24

- If we wish to implement bgtz (branch greater than zero) how would we change our CPU?
Review Problem 25

- What mods are needed to support jump register:
  - PC = Reg[RS]
Review Problem 26

- Show the control settings needed to implement “addi $rt, $rs, imm16”
Review Problem 27

- 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?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RegDst</td>
<td></td>
</tr>
<tr>
<td>ALUSrc</td>
<td></td>
</tr>
<tr>
<td>MemToReg</td>
<td></td>
</tr>
<tr>
<td>RegWr</td>
<td></td>
</tr>
<tr>
<td>MemWr</td>
<td></td>
</tr>
<tr>
<td>Branch</td>
<td></td>
</tr>
<tr>
<td>Jump</td>
<td></td>
</tr>
<tr>
<td>ALUCntrl</td>
<td></td>
</tr>
</tbody>
</table>
Review Problem 28

- When we discussed inserting registers, we limited it to Acyclic Combinational Logic. Why?
Review Problem 29

- 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 30

- 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 31

- 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 32

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

```
add $1, $2, $3
nor $4, $5, $6
sub $7, $8, $9
slt $10, $11, $12
nand $13, $14, $15
```
Review Problem 33

- The MIPS have 2-operand branches that test equality/inequality, and 1-operand branches that test greater/equal/less than 0, but no branches that test greater/less than for 2 operands. Why?
Review Problem 34

- Do the jump instructions (j, jr) have problems with hazards?
Review Problem 35

- What forwarding happens on the following code?

```assembly
lw $t0, 0($t1)
add $t2, $t3, $t3
nor $0, $t0, $t4
bne $t2, $0, END
sub $t5, $t2, $t4
```
Review Problem 36

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

```assembly
and $t0, $t1, $t2
ori $t0, $t0, 7
add $t3, $t4, $t5
lw $t6, 0($t3)
bgez $t6, FOO
j BAR
```
Review Problem 37

- 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 38

- 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 39

- The length (number of blocks) in a direct mapped cache is always a power of 2. Why?
Review Problem 40

- 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 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]++;
```
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: lw $t0, 0($s0)
2: lw $t1, 0($s1)
3: sw $t1, 0($s0)
4: sw $t0, 0($s1)
```
Review Problem 51

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

1: lw $t1, 4($a0)
2: add $t2, $t1, $a0
3: lw $t3, 8($a1)
4: sub $t4, $t3, $a2
5: sw $t5, 0($a3)
6: beq $s0, $s1, FOO
Review Problem 52

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

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

- $s0$-$s1$ hold two values. Set $s0$ to the smaller of the two, and $s1$ to the larger, 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?
Review Problem 56

- 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 MIPS instructions except multiply and divide, but must use only Intels’s 2 operand formats. Compute $t1 = t0*5$. If necessary, you can use a MOVE instruction, that copies one register to another.