Kyler Little
CPT_S 260

Homework #1

1. What was the name of the first computer? With how much memory? How much memory do you have in your laptop? in your cellphone?

If you simply require that a computer be a physical device that implements an instruction set and stores memory, then a human could technically be a computer, as we've discussed in class. So in reality, this question depends on your definition of computer. The first recognized device for computing was the abacus, invented in Mesopotamia around 2500 BCE. It stored the numbers it added and subtracted in the form of beads, so that was its memory. The first general-purpose computer was the Analytical Engine, which could store 1000 forty-decimal-digit numbers (about 16 KB). Taking it one step further, the first electromechanical programmable, fully automatic digital computer was Germany's Z3. The Z3 could store 64 22-bit words for data memory (176 bytes).

Weirdly enough, both my computer and cell phone have 128 GB of memory.

2. In 1974, Intel 8080 processor was released with clock rate roughly 2Mhz. Nowadays, you can buy Intel Pentium D 830 with 3Ghz clock rate for 20 dollars, or if you have a little more money to spend, an Intel i7 with 4-core and 4Ghz clock rate costs about 300 dollars. Roughly, how many times faster do the processors get today?

The answer to this question depends on the type of instructions fed to the processor. If the instructions are perfectly parallelizable (i.e. each line can be run at the same time without corruption of the data), then multiple cores will allow the processor to run the instructions more quickly than a single processor could, even if other variables were kept exactly the same. On the other hand, if the instructions fed to the processor were linear, then having multiple cores wouldn't affect anything. For this problem, I will assume that the instructions are linear. In that case, we simply must calculate the difference in processor rate: 3Ghz / 2Mhz = 1500x faster or 4Ghz / 2Mhz = 2000x faster.

3. It is known that, using comparisons, one can sort five numbers in seven comparisons. Please give me the instruction set on which one can write such a program to sort five numbers.

The algorithm that does this is merge sort. Thus, I will create an instruction set for merge sort. At a high level, the algorithm requires dividing an array into two, merging two sorted arrays into one, and comparisons. At a lower level, these operations can be achieved by operations such as branching, comparing, goto, divide, sizeof, etc. I list the operations required on the next page. I'm not sure if this covers all of the necessary instructions, but I believe it covers most all of them.

```
// cmp (compare): given an array A, compare A[i] and A[j] according to operator o and store the boolean result in r
//                o can be >, <, !=, ==
//                to actually perform the comparison, cmp will call an arithmetic function like add or subtract
//                for example, if the comparison is A[i] > A[j], then we call subtract(A[i], A[j], r)
//                if r > 0, then assign r = true; else, r = false; this can be implemented using branch function
cmp(A, i, j, o, r)

// branch: if condition is not zero, then go to address a1, else go to address a2
branch(c, a1, a2)

// goto: go to the address a in memory
goto(a)

// size: returns the size of the array between indices i and j and stores the result in r
//       this is useful for mergesort because we are at the base case when size() returns 1 (a sorted array)
size(A, i, j, r)

// add: add L and R and store the result in r
add(L, R, r)

// subtract: subtract R from L and store the result in r
subtract(L, R, r)

// >: bit shift the number n to the right, effectively dividing n by 2
//    useful for merge sort
>(n, r)

// return: this function will pop the current stack's pointer and goto to the most recent pointer on stack
return()

// merge: at a high level, it merges sorted arrays A and B into a sorted array C
//        uses cmp to do so
merge(A, B, C)
```