Midterm CheatSheet:

1. Assume that $s0 to $s4 hold inteers and may be used as indices or offsets. Assume that the base addresses of arrays A and B are in $s6 and $s7. $t registers are used to hold temporary values. What does this code do?

(add)Add $t0, $s6, $s0 – Temp0  = A + s0

(add)Add $t1, $s7, $s1 – Temp1 = B + s1

(load)lw $s0, O($t0) – load s0 = temp0 -> load s0 = (A + s0)

(iadd)addi $t2, $t0, 4 -- temp2 = temp0 + 4 -> temp2 = (A + s0 + 4)

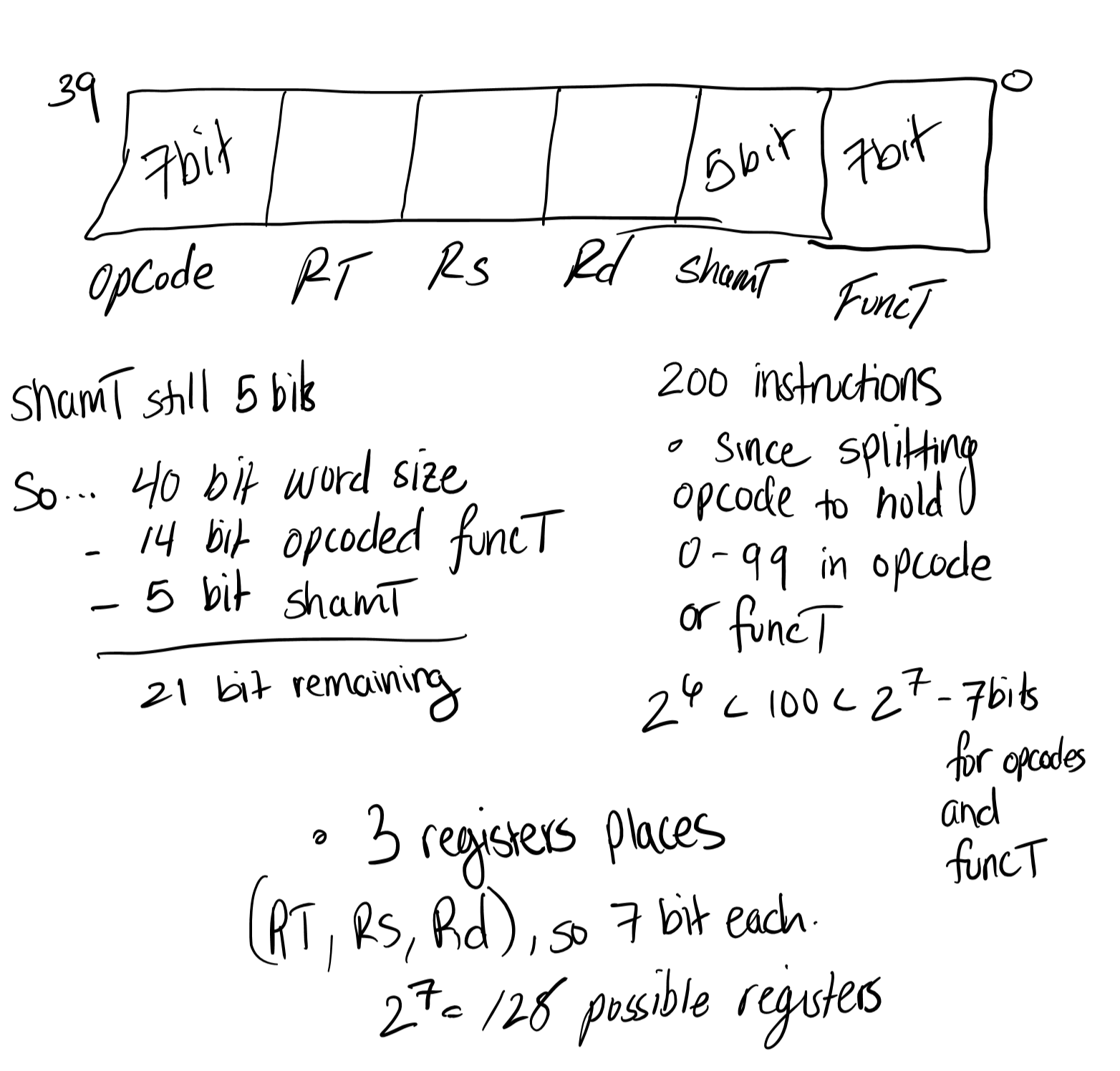
(load)lw $t0, 0($t2) -- load temp0 = temp2 -> temp0 = ( A + s0 + 4)

(add)add $t0, $t0, $s0

(store)sw $t2, O($t1) -- store temp2 = temp1 -> temp2 = (B + s1)

* This code stores three ints in $s0, $t0, and $t2. Initially starts by setting the index to store at $t0 and $t1, then stores the value at A[INDEX] in $s0, It then increments the index by the size of an int (4 bytes), and then stores that value, at A[index], into $t0. It then adds the contents of $t0 and $s0, which are A[index] + A[index] and stores the result in $t0. $t0 is then stored in register $t2; Therefore, $t0 = A[index] + A[index].

1. You are tasked with designing a CPU with 40-bit wordsize. You are using a RISC instruction set, similar to MIPS, with about 200 instruction codes. How many registers do you have? Why?

* 27  = 128 registers. 

1. On a 32bit MIPS machine, you need to execute the following single-precision floating point arithmetic instructions.
   1. 3.5e25 \* 3.5e25 / 12.0e20 -> Has overflow, the denominator is too small
   2. 13.2e-30 \* 3.5e25 -> does not have overflow or underflow
   3. 13.2e-20 / 3.5e25 + 5.7e8 -> has underflow’ The numerator exponent needs to be positive.

Do any of the above operations overflow and underflow? If so, can they be rewritten so they work? Note: all the above expressions are in base 10, the notation eNNN means 10 raised to the NNN Power.

