1 memset (the pointer arithmetic way...)

In lecture, we discussed pointers and how they are just memory addresses at the machine level. Below is C code for a function that writes an integer value repeatedly `num_words` times starting from address `dst`.

```
void memset(int *dst, int value, int num_words) {
    for (int i = 0; i < num_words; ++i) {
       *dst = value;
        dst++;
    }
}
```

Write it as a MIPS function:
Solution

```
.text
memset:
        li      $t1, 0          # i: $t1, dst: $a0, value: $a1, num_words: $a2
loop:   bge     $t1, $a2, exit  # If i >= num_words, exit loop
        sw      $a1, 0($a0)     # *dst = value;
        addi    $a0, $a0, 4     # Note address increment by 4!
        addi    $t1, $t1, 1     # i++;
        j       loop

exit:
        jr      $ra
```

2 Pointers and Structures

```
void increment(node_t * head, int value) {
  for(node_t * trav = head; trav != NULL; trav = trav->next) {
    *(trav->data) += value;
  }
}
```

Write increment as a MIPS function.

```
increment:
      # $a0 = head (of type node_t *), $a1 = value (of type int)
increment_loop:
      beq $a0, $zero, increment_done      # if( a0 == NULL ) exit
      lw $t0, 0($a0)                      # t0 = (*a0).data , i.e. t0 = a0->data
      lw $t1, 0($t0)                      # t1 = *(t0) , i.e. t1 = *(a0->data)
      add $t1, $t1, $a1                   # t1 += value
      sw $t1, 0($t0)                      # *(a0->data) = t1
      lw $a0, 4($a0)                      # a0 = (*a0).next , i.e. a0 = a0->next
      j loop
increment_done:
      jr $ra
```