a4-part-1.asm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Terms: | set\_16x16 (Test 1a) | set\_16x16 (Test 1b) | aggregate | get\_16x16 (Test 1c) | copy\_16x16 |
| Total | 23 | 23 | 46 | 23 | 1807 |
| ALU | 11 | 11 | 22 | 10 | 777 |
| Jump | 2 | 2 | 4 | 2 | 2 |
| Branch | 4 | 4 | 8 | 5 | 513 |
| Memory | 1 | 1 | 2 | 1 | 514 |
| Other | 5 | 5 | 10 | 5 | 1 |

a4-part-2.asm:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Terms: | sum\_neighbours (Test 2a) | sum\_neighbours (Test 2b) | sum\_neighbours (Test 2c) | aggregate |
| Total | 296 | 308 | 262 | 866 |
| ALU | 95 | 101 | 91 | 287 |
| Jump | 18 | 18 | 18 | 54 |
| Branch | 49 | 52 | 34 | 135 |
| Memory | 101 | 104 | 99 | 304 |
| Other | 33 | 33 | 20 | 86 |

a4-part-3.asm:

|  |  |
| --- | --- |
| Terms: | bitmap\_to\_16x16 |
| Total | 25034 |
| ALU | 14208 |
| Jump | 1541 |
| Branch | 4128 |
| Memory | 2852 |
| Other | 2305 |

In a4-part-2.asm, the procedure sum\_neighbours is used to sum the elements relative to the given row and column. In order to get the values of the neighbouring array elements we are to call get\_16x16 four times (for the top row, right column, bottom row, and the left column). First and foremost, there are clear more ALU instructions because more arithmetic and logical operations are needed or sum\_neighbours. There are instructions which are repeated because of loops and other registers are used for adjusting rows and columns ($a1 and $a2), incrementing the counter ($s1), and adding to the sum total of neighbouring values ($s0). These all lead to sum\_neighbours having a higher number of ALU instructions compared to get\_16x16. Secondly, there are more jump instructions in a4-part-2.asm for several reasons. At the beginning stages of the program there is a jump and link call to sum\_neighbours. Then, there are several repeating jal calls to get\_16x16 in each loop to get the values from neighbouring elements. Thirdly, branch instructions in sum\_neighbours are used to check if all the neighbouring elements are read and to avoid going out of range to check for elements. Moreover, after calling get\_16x16, the procedure check to see if the given row and column is valid (0 <= row < 16 and 0 <= column <16) and branches out of get\_16x16 if the given index is not valid. The use of memory is very different in sum\_neighbours compared to get\_16x16. The procedure sum\_neighbours must preserve the values for $ra, $s0, $s1, $a0, $a1, and $a2 before jump and linking to get\_16x16. This means that a lot of memory must be used to use a stack to push these registers and then shrink the stack by restoring the original values. There is a very low use of memory in get\_16x16 because the only instance of use is to load the byte from the given row and column in the array.