

Samphire Control Flow Instructions: Jumps

- The Unconditional Jump Instruction

`jmp label`

opcode ↗ ↖ This refers to an Identifier which is associated with another instruction in the Program

116

Samphire Control Flow Instructions: Jumps

- Thus:

```
Instruction1
instruction2
label:
Instruction3
Instruction4
jmp label
```

117

Samphire Control Flow Instructions: Jumps

- The `jmp label` instruction is translated by the assembler into:
`C0 number`

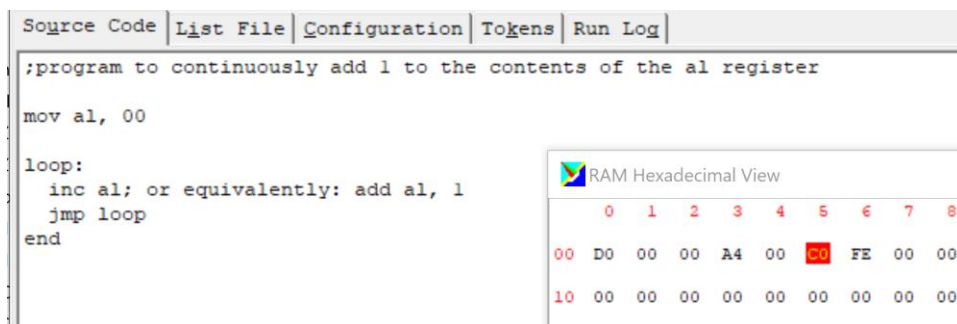
Where `number` is the number of bytes between the `jmp label` instruction and the instruction identified by the label (in this case `Instruction3`)

- If the `label` occurs earlier in the code then the `jmp label` instruction (as in the example above), the `number` is negative, otherwise it is positive.
- The machine executes the `jmp` instruction by adding the calculated `number` to the value of the Instruction Pointer (IP)

118

Samphire Control Flow Instructions: Jumps

- Consider the following code that continuously adds 1 to the contents of the `al` register:
 - Notice that the register overflows back to 0 when 1 is added to FF



The screenshot shows the Samphire IDE with the following assembly code in the Source Code window:

```

;program to continuously add 1 to the contents of the al register

mov al, 00

loop:
  inc al; or equivalently: add al, 1
  jmp loop
end
  
```

Overlaid on the bottom right is the RAM Hexadecimal View window, which displays the memory state:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|----|----|----|----|----|----|----|----|----|
| 00 | D0 | 00 | 00 | A4 | 00 | C0 | FE | 00 | 00 |
| 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

119

Samphire Control Flow Instructions: Jumps

- Other jump instructions include the conditional jmp instructions:
 - jz label ; jump to the label if the zero flag is set
 - jnz label ; jump to the label if the zero flag is not set
 - js label ; jump to the label if the sign flag is set
 - jns label ; jump to the label if the sign flag is not set
 - jo label ; jump to the label if the overflow flag is set
 - jno label ; jump to the label if the overflow flag is not set

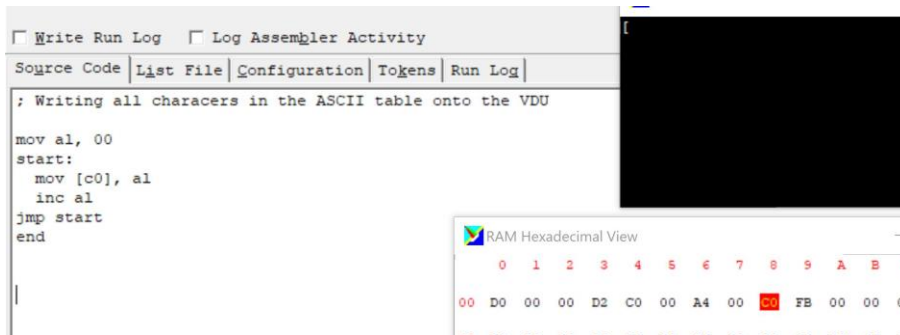
120

Memory Mapped I/O

- The portion of the Samphire RAM from C0 to FF is associated with the Visual Display Unit (VDU).
- Any values placed in these memory locations are interpreted as ASCII values and they are displayed on the VDU.

121

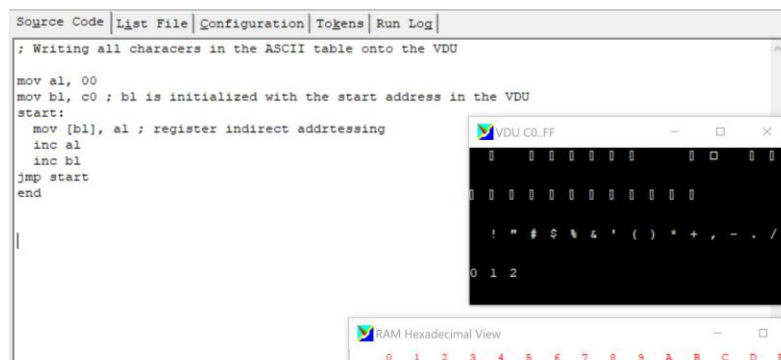
Writing on the VDU



122

Writing on the VDU

- If we wish to write to multiple locations in the VDU, a good strategy is to keep the address of the next VDU location in a register:
 - Here the bl register is used to hold the address of the next DVU location. It is initialized to C0 and is incremented by 1 each time through to loop



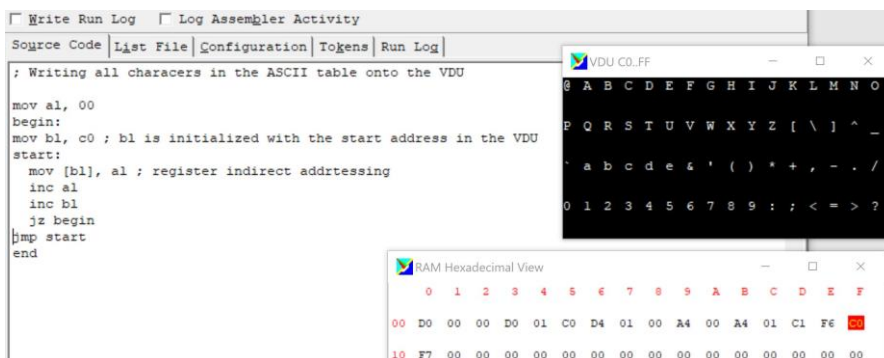
123

Writing on the VDU

- There is a problem with this code: when the value in the bl register reaches FF, it will be incremented and will overflow to 00.
- When this happens we will start to write into memory at address 00 and will overwrite our program
- To confine the writing to the VDU, we must reset it to C0 when we reach the end of the VDU RAM.
- We do this use the conditional jump instruction `jz` to jump to a different label if the zero flag is set
- The zero flag is set when the result of the previous instruction is zero (0), and indeed when bl is incremented to yield zero, the flag will be set.

124

Writing on the VDU



Note that we have now introduced a second label.

Lots of labels and jumping around the place can quickly make your program difficult to read and understand.

125