|  |  |
| --- | --- |
| i = 0  if (i < 3):  j = 10  while (j > 0):  i = i + 1  j = j – 1  i = i + i  print i | The source code here has been compiled into the assembly code below. The compiler will generally try to keep variables in a single register. Here the compiler uses R0 for i and R2 for j. |

## Exercise #1: Source Code to Assembly

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Addr | Binary/Hex | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Disassembly | Comment |
| 000 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | MOVI R0, 0 | *Set i = 0* |
| 1 | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 0 | | | | | |
| 002 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | MOVI R1, 3 |  |
| 1 | | | | | | | | 1 | | | | | | | | 0 | | | | | | | | 3 | | | | | |
| 004 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | CMP R0, R1 |  |
| 4 | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 1 | | | | | |
| 006 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | JGTE +10 (01A) | *Jump if R0 >= R1 (i >=3)* |
| 5 | | | | | | | | 4 | | | | | | | | 0 | | | | | | | | A | | | | | |
| 008 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | MOVI R2, 0xA | *Set j = 10* |
| 1 | | | | | | | | 2 | | | | | | | | 0 | | | | | | | | A | | | | | |
| 00A | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | MOVI R3, 0 |  |
| 1 | | | | | | | | 3 | | | | | | | | 0 | | | | | | | | 0 | | | | |
| 00C | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | CMP R2, R3 |  |
| 4 | | | | | | | | 0 | | | | | | | | 2 | | | | | | | | 3 | | | | |
| 00E | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | JLTE +5 (018) | *Jump if R2 <= R3 (j <= 0)* |
| 5 | | | | | | | | 3 | | | | | | | | 0 | | | | | | | | 5 | | | | |
| 010 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | MOVI R4, 1 |  |
| 1 | | | | | | | | 4 | | | | | | | | 0 | | | | | | | | 1 | | | | |
| 012 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | ADD R0, R0, R4 | *Set i = i + 1* |
| 2 | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 4 | | | | |
| 014 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | SUB, R2, R2, R4 | *Set j = j – 1* |
| 3 | | | | | | | | 2 | | | | | | | | 2 | | | | | | | | 4 | | | | |
| 016 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | JMP 00C | *Loop* |
| 6 | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | C | | | | |
| 018 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | ADD R0, R0, R0 | *Set i = i + i* |
| 2 | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 0 | | | | |
| 01A | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | OUT R0 | *Print i* |
| B | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 0 | | | | |

## Exercise 2: Assembly to Source Code

Translate the assembly below into C/Python/pseudocode. What algorithm does this code represent? What would this code output?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Binary / Hex | | | | | | | | | | | | | | | | Disassembly | Comment |
| 000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | MOVI R1, 1 |  |
| 1 | | | | 1 | | | | 0 | | | | 1 | | | |
| 002 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | MOVI R2, 1 |  |
| 1 | | | | 2 | | | | 0 | | | | 1 | | | |
| 004 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ADD R3, R1, R2 |  |
| 2 | | | | 3 | | | | 1 | | | | 2 | | | |
| 006 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | OUT R3 |  |
| B | | | | 0 | | | | 0 | | | | 3 | | | |
| 008 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | MOV R1, R2 |  |
| 0 | | | | 0 | | | | 1 | | | | 2 | | | |
| 00A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | MOV R2, R3 |  |
| 0 | | | | 0 | | | | 2 | | | | 3 | | | |
| 00C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | JMP 000 |  |
| 7 | | | | 0 | | | | 0 | | | | 0 | | | |

## Exercise 3: Crack the Code

The code below is a simple decryption routine recovered from a crypto device. Can you recover the algorithm and decrypt the message? The following message was captured along with the code:

**L W V B O Q D M C X B P M A P Q X (12,23,22,2,15,17,4,13,3,24,2,16,13,1,16,17,24)**

NOTE: For the purposes of this exercise, assume numbers are represented as 1-26, or 1-0x1A in hexadecimal. Normally in electronics the ASCII format is used, where letters are represented as 0x41-0x5A (upper case) and 0x61 to 0x7A (lower case), but we are simplifying it a bit.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Binary / Hex | | | | | | | | | | | | | | | | Disassembly | Comment |
| 000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IN R1 |  |
| A | | | | 0 | | | | 0 | | | | 1 | | | |
| 002 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | MOVI R2, 0x12 |  |
| 1 | | | | 2 | | | | 1 | | | | 2 | | | |
| 004 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ADD R1, R1, R2 |  |
| 2 | | | | 1 | | | | 1 | | | | 2 | | | |
| 006 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | MOVI R2, 0x1A |  |
| 1 | | | | 2 | | | | 1 | | | | A | | | |
| 008 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | CMP R1, R2 |  |
| 4 | | | | 0 | | | | 1 | | | | 2 | | | |
| 00A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | JGT +2 (00E) |  |
| 5 | | | | 2 | | | | 0 | | | | 2 | | | |
| 00C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | SUB R1, R1, R2 |  |
| 3 | | | | 1 | | | | 1 | | | | 2 | | | |
| 00E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | OUT R1 |  |
| B | | | | 0 | | | | 0 | | | | 1 | | | |
| 010 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | JMP 000 |  |
| 7 | | | | 0 | | | | 0 | | | | 0 | | | |

## Exercise 4: Find the Bug

You have a program that accepts the hour on the 24-hour clock as two digits (two registers) and outputs the hour on a 12-hour clock (one register).

|  |  |  |
| --- | --- | --- |
| INPUT | OUTPUT | First Instructions |
| “14” | “02” | R1 = 1, R2 = 4 |
| “09” | “09” | R1 = 0, R2 = 9 |
| “23” | “11” | R1 = 2, R2 = 3 |

Determine the algorithm for this program. This program includes at least one error. Can you find it? Once you identify the error(s), how would you patch the program to fix the errors?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Addr | Binary/Hex | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Disassembly | Comment |
| 000 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | IN R1 |  |
| A | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 1 | | | | | |
| 002 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | IN R2 |  |
| A | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 2 | | | | | |
| 004 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | MOVI R3, 0xA |  |
| 1 | | | | | | | | 3 | | | | | | | | 0 | | | | | | | | A | | | | | |
| 006 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | MUL R1, R1, R3 |  |
| C | | | | | | | | 1 | | | | | | | | 1 | | | | | | | | 3 | | | | | |
| 008 |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | ADD R2, R1, R2 |  |
| 2 | | | | | | | | 2 | | | | | | | | 1 | | | | | | | | 2 | | | | | |
| 00A | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | MOVI R3, 0xC |  |
| 1 | | | | | | | | 3 | | | | | | | | 0 | | | | | | | | C | | | | |
| 00C | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | MOVI R5, 0 |  |
| 1 | | | | | | | | 5 | | | | | | | | 0 | | | | | | | | 0 | | | | |
| 00E | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | CMP R2, R5 |  |
| 4 | | | | | | | | 0 | | | | | | | | 2 | | | | | | | | 5 | | | | |
| 010 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | JNE +2 (014) |  |
| 5 | | | | | | | | 5 | | | | | | | | 0 | | | | | | | | 2 | | | | |
| 012 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | MOV R2, R4 |  |
| 0 | | | | | | | | 0 | | | | | | | | 2 | | | | | | | | 4 | | | | |
| 014 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | CMP R2, R4 |  |
| 4 | | | | | | | | 0 | | | | | | | | 2 | | | | | | | | 4 | | | | |
| 016 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | JLTE +2 (01A) |  |
| 5 | | | | | | | | 3 | | | | | | | | 1 | | | | | | | | A | | | | |
| 018 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | SUB R2, R2, R4 |  |
| 3 | | | | | | | | 2 | | | | | | | | 2 | | | | | | | | 4 | | | | |
| 01A | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |  | OUT R2 |  |
| B | | | | | | | | 0 | | | | | | | | 0 | | | | | | | | 2 | | | | |