**Hack Computer**

To convert the provided assembly language programs into machine language for the Hack computer, I followed these steps:

1. **Understand the Assembly Instructions**: I carefully read and understood each assembly instruction in the given programs to determine their functionality and purpose.
2. **Translate Symbols to Binary**: I converted each symbolic representation (e.g., **@**, **D**, **M**, **+**, **-**, **JGT**, **JMP**) into their respective binary representations based on the Hack architecture specifications.
3. **Write Binary Instructions**: Using the binary representations of symbols, I wrote down the binary instructions for each assembly instruction in the correct order.
4. **Load and Execute**: Finally, I loaded the generated machine language programs into a Hack CPU simulator to execute and verify their correctness.

**Program 1: Computes R0 = 2 + 3**

**Step 1: Break Down the Assembly Instructions**

* @2: Load the constant 2 into the A register.
* D=A: Store the value from the A register into the D register.
* @3: Load the constant 3 into the A register.
* D=D+A: Add the value in the A register to the value in the D register.
* @0: Load the address 0 into the A register.
* M=D: Store the value from the D register into the memory address specified by the A register.

**Step 2: Convert Symbols to Binary**

* @2: Binary representation of 2 is 0000000000000010.
* D=A: C instruction format: 111accccccdddjjj, where a=0, c=101010, d=000, and j=000.
* @3: Binary representation of 3 is 0000000000000011.
* D=D+A: C instruction format: 111accccccdddjjj, where a=0, c=000010, d=000, and j=000.
* @0: Binary representation of 0 is 0000000000000000.
* M=D: C instruction format: 111accccccdddjjj, where a=1, c=100000, d=000, and j=000.

**Assembler Program:**

// Computes R0 = 2 + 3

@2

D=A

@3

D=D+A

@0

M=D

**Machine Language Program**:

// Computes R0 = 2 + 3

// Line 1: @2

0000000000000010

// Line 2: D=A

1110110000010000

// Line 3: @3

0000000000000011

// Line 4: D=D+A

1110000010010000

// Line 5: @0

0000000000000000

// Line 6: M=D

1110001100001000

**Program 2: Symbol-less version of the Max.asm program**

**Assembler Program**:

// Symbol-less version of the Max.asm program

@0

D=M

@1

D=D-M

@10

D;JGT

@1

D=M

@12

0;JMP

@0

D=M

@2

M=D

@14

0;JMP

**Machine Language Program**:

// Symbol-less version of the Max.asm program

// Line 1: @0

0000000000000000

// Line 2: D=M

1111110000010000

// Line 3: @1

0000000000000001

// Line 4: D=D-M

1111010011010000

// Line 5: @10

0000000000001010

// Line 6: D;JGT

1110001100000001

// Line 7: @1

0000000000000001

// Line 8: D=M

1111110000010000

// Line 9: @12

0000000000001100

// Line 10: 0;JMP

1110101010000111

// Line 11: @0

0000000000000000

// Line 12: D=M

1111110000010000

// Line 13: @2

0000000000000010

// Line 14: M=D

1110001100001000

// Line 15: @14

0000000000001110

// Line 16: 0;JMP

1110101010000111

Through this process, I gained a deeper understanding of how assembly language instructions are translated into binary machine language instructions. It reinforced the importance of understanding the underlying hardware architecture and the relationship between software and hardware in computing systems.