### **1. Program Constants**

assembly

Copy

CR EQU $0D \* Carriage return  
LF EQU $0A \* Line feed

* These constants represent ASCII values:
  + CR = $0D (Carriage Return)
  + LF = $0A (Line Feed)

These are used for formatting output by inserting line breaks.

### **2. Start of Program**

assembly

Copy

ORG $1000  
START:

* ORG $1000: The program starts at memory address $1000.
* START: is a label marking the program's entry point.

### **3. Display Welcome Message**

assembly

Copy

LEA WELCOME\_MSG,A1  
 MOVE.B #14,D0  
 TRAP #15

* LEA WELCOME\_MSG, A1: Loads the address of the WELCOME\_MSG string into register A1.
* MOVE.B #14, D0: Specifies the length of the string (14 bytes).
* TRAP #15: Calls a monitor service to display the string on the screen.

### **4. Input First Number**

assembly

Copy

INPUT\_NUM1:  
 LEA PROMPT\_NUM1,A1  
 MOVE.B #14,D0  
 TRAP #15  
   
 MOVE.B #4,D0  
 TRAP #15  
 MOVE.L D1,D5 \* Store first number in D5

* Displays the prompt asking for the first number (PROMPT\_NUM1).
* Reads the number from the user and stores it in register D5.

### **5. Input Second Number**

assembly

Copy

INPUT\_NUM2:  
 LEA PROMPT\_NUM2,A1  
 MOVE.B #14,D0  
 TRAP #15  
   
 MOVE.B #4,D0  
 TRAP #15  
 MOVE.L D1,D6 \* Store second number in D6

* Prompts the user for the second number (PROMPT\_NUM2).
* Reads the second number from the user and stores it in register D6.

### **6. Input Operation**

assembly

Copy

INPUT\_OP:  
 LEA PROMPT\_OP,A1  
 MOVE.B #14,D0  
 TRAP #15  
   
 MOVE.B #5,D0 \* Change to read characters  
 TRAP #15  
   
 SUB.B #'0',D1 \* Convert ASCII to numeric value

* Displays the operation prompt (PROMPT\_OP), listing all available operations.
* Reads the operation character (1-6) and converts it from ASCII to a numeric value using SUB.B #'0', D1.

### **7. Check Operation Range**

assembly

Copy

CMP.B #1,D1 \* Check if less than 1  
 BLT INVALID\_OP  
 CMP.B #6,D1 \* Check if greater than 6  
 BGT INVALID\_OP

* Ensures the operation input is between 1 and 6 (valid operations). If not, it jumps to the INVALID\_OP label.

### **8. Perform Calculation**

assembly

Copy

CALCULATE:  
 CMP.B #1,D1 \* Addition  
 BEQ ADD\_NUMS  
 CMP.B #2,D1 \* Subtraction  
 BEQ SUB\_NUMS  
 CMP.B #3,D1 \* Multiplication  
 BEQ MUL\_NUMS  
 CMP.B #4,D1 \* Division  
 BEQ DIV\_NUMS  
 CMP.B #5,D1 \* Modulus  
 BEQ MOD\_NUMS  
 CMP.B #6,D1 \* Percentage  
 BEQ PERC\_NUMS  
   
 \* Invalid operation handler  
 BRA INVALID\_OP

* Compares the operation input (stored in D1) to determine which subroutine to call (addition, subtraction, multiplication, division, modulus, or percentage). If no valid operation is matched, it jumps to INVALID\_OP.

### **9. Arithmetic Subroutines**

Each arithmetic operation (addition, subtraction, multiplication, division, modulus, and percentage) is implemented in its own subroutine.

#### **Addition:**

assembly

Copy

ADD\_NUMS:  
 MOVE.L D5,D1  
 ADD.L D6,D1  
 BRA DISPLAY\_RESULT

* Copies the first number (D5) into D1 and adds the second number (D6) to D1.
* Then, it branches to DISPLAY\_RESULT.

#### **Subtraction:**

assembly

Copy

SUB\_NUMS:  
 MOVE.L D5,D1  
 SUB.L D6,D1  
 BRA DISPLAY\_RESULT

* Similar to addition but performs subtraction instead. After that, it branches to DISPLAY\_RESULT.

#### **Multiplication:**

assembly

Copy

MUL\_NUMS:  
 MOVE.L D5,D1  
 MULS D6,D1  
 BRA DISPLAY\_RESULT

* Multiplies the first number (D5) by the second (D6) and stores the result in D1, then branches to DISPLAY\_RESULT.

#### **Division:**

assembly

Copy

DIV\_NUMS:  
 \* Check for division by zero  
 CMP.L #0,D6  
 BEQ DIV\_ZERO  
   
 MOVE.L D5,D1  
 DIVS D6,D1  
 BRA DISPLAY\_RESULT

* First checks if the second number (D6) is zero to avoid division by zero. If not zero, it divides D5 by D6 and stores the result in D1. Then it branches to DISPLAY\_RESULT.

#### **Modulus:**

assembly

Copy

MOD\_NUMS:  
 \* Check for division by zero  
 CMP.L #0,D6  
 BEQ DIV\_ZERO  
   
 \* Modulus calculation: D1 = D5 % D6  
 MOVE.L D5,D1  
 DIVS D6,D1  
 SWAP D1 \* Remainder is in upper word  
 EXT.W D1 \* Sign extend the remainder  
 EXT.L D1 \* Extend to long word  
 BRA DISPLAY\_RESULT

* Checks for division by zero. If the second number is non-zero, it performs the modulus operation (D5 % D6) by dividing D5 by D6 and using the remainder, which is swapped into the upper word and sign-extended.

#### **Percentage:**

assembly

Copy

PERC\_NUMS:  
 \* Calculate percentage: (D5 / D6) \* 100  
 \* Check for division by zero  
 CMP.L #0,D6  
 BEQ DIV\_ZERO  
   
 \* Percentage calculation  
 MOVE.L D5,D1  
 MULS #100,D1 \* Multiply first number by 100  
 DIVS D6,D1 \* Divide by second number  
 BRA DISPLAY\_RESULT

* Checks for division by zero, and if valid, calculates the percentage by multiplying D5 by 100 and then dividing by D6.

### **10. Display Result**

assembly

Copy

DISPLAY\_RESULT:  
 \* Display result message  
 LEA RESULT\_MSG,A1  
 MOVE.B #14,D0  
 TRAP #15  
   
 \* Convert and display numeric result  
 MOVE.B #3,D0  
 TRAP #15  
 \* New line  
 LEA NEWLINE,A1  
 MOVE.B #14,D0  
 TRAP #15  
 BRA EXIT

* Displays the result message (RESULT\_MSG).
* Displays the result stored in D1.
* Then, it displays a newline (NEWLINE) and exits the program.

### **11. Error Handling**

#### **Invalid Operation:**

assembly

Copy

INVALID\_OP:  
 LEA ERR\_OP\_MSG,A1  
 MOVE.B #14,D0  
 TRAP #15  
 BRA EXIT

* If an invalid operation is entered, it displays the error message (ERR\_OP\_MSG) and exits the program.

#### **Division by Zero:**

assembly

Copy

DIV\_ZERO:  
 LEA ERR\_DIV\_ZERO,A1  
 MOVE.B #14,D0  
 TRAP #15

* If division by zero is attempted, it displays the division-by-zero error message (ERR\_DIV\_ZERO).

### **12. Program Exit**

assembly

Copy

EXIT:  
 MOVE.B #9,D0  
 TRAP #15

* Exits the program by calling a monitor service with exit code #9.

### **13. Data Section**

assembly

Copy

WELCOME\_MSG DC.B 'Extended Calculator Program',CR,LF,0  
PROMPT\_NUM1 DC.B 'Enter first number: ',0  
PROMPT\_NUM2 DC.B 'Enter second number: ',0  
PROMPT\_OP DC.B 'Enter operation:',CR,LF  
 DC.B '1. Addition',CR,LF  
 DC.B '2. Subtraction',CR,LF  
 DC.B '3. Multiplication',CR,LF  
 DC.B '4. Division',CR,LF  
 DC.B '5. Modulus',CR,LF  
 DC.B '6. Percentage',CR,LF  
 DC.B 'Choose operation (1-6): ',0  
RESULT\_MSG DC.B 'Result: ',0  
NEWLINE DC.B CR,LF,0  
ERR\_OP\_MSG DC.B 'Invalid operation!',CR,LF,0  
ERR\_DIV\_ZERO DC.B 'Error: Division by zero!',CR,LF,0

* This section defines the messages used throughout the program: welcome message, prompts for the numbers and operation, result message, error messages, and formatting (newlines).