MM Algorithms (use start and stop step)

8 bit and 16 bit arth
Note: Replace `AL` and `BL` with `AX` and `BX` in 16-bit operations.

(A) 8-Bit Addition

1. Load first number into AL.
2. Load second number into BL.
3. Add BL to AL.
4. Store the result in memory.

(B) 8-Bit Subtraction

1. Load first number into AL.
2. Load second number into BL.
3. Subtract BL from AL.
4. Store the result in memory.

(C) 8-Bit Multiplication

1. Load first number into AL.
2. Load second number into BL.
3. Multiply AL by BL.
4. Store the result in memory.

(D) 8-Bit Division

1. Load dividend into AL.
2. Load divisor into BL.
3. Divide AL by BL.
4. Store quotient in memory and remainder in another location.

(E) 16-Bit Addition

1. Load first number into AX.
2. Load second number into BX.
3. Add BX to AX.
4. Store the result in memory.

(F) 16-Bit Subtraction

1. Load first number into AX.
2. Load second number into BX.
3. Subtract BX from AX.
4. Store the result in memory.

(G) 16-Bit Multiplication

1. Load first number into AX.
2. Load second number into BX.
3. Multiply AX by BX.
4. Store the result (lower part) in memory and higher part in another location.

(H) 16-Bit Division

1. Load dividend into AX.
2. Load divisor into BX.
3. Divide AX by BX.
4. Store quotient in memory and remainder in another location.

(I) 16-Bit Addition with Carry

1. Load first number into AX.
2. Load second number into BX.
3. Initialize carry register (CX) to 0.
4. Add BX to AX.
5. If carry occurs, increment CX.
6. Store result and carry in memory.

(J) 16-Bit Subtraction with Borrow

1. Load first number into AX.
2. Load second number into BX.
3. Initialize borrow register (CX) to 0.
4. Subtract BX from AX.
5. If borrow occurs, increment CX and adjust AX.
6. Store result and borrow in memory.

Bit Manipulation

(A) Bit Complement

1. Load value from memory into AX.
2. Perform bitwise NOT on AX.
3. Store the result in memory.

(B) Bit Setting

1. Load value from memory into AX.
2. Load mask from memory into BX.
3. Perform bitwise OR between AX and BX.
4. Store the result in memory.

(C) Bit Meshing

1. Load value from memory into AX.
2. Load mask from memory into BX.
3. Perform bitwise AND between AX and BX.
4. Store the result in memory.

Bubble Sort

(A) Bubble Sort - Ascending

1. Load the number of elements into register CH.
2. Decrement CH to set up for sorting.
3. Set the outer loop counter to the number of elements.

4. Initialize index SI to the start of the array.
5. Compare each element with the next:
   - If the current element is greater than the next, swap them.
6. Repeat until the entire array is sorted.

(B) Bubble Sort - Descending

1. Load the number of elements into register CH.
2. Decrement CH to set up for sorting.
3. Set the outer loop counter to the number of elements.
4. Initialize index SI to the start of the array.
5. Compare each element with the next:
   - If the current element is less than the next, swap them.
6. Repeat until the entire array is sorted.

Stepper motor

1. Initialize \`DI\` to 1200 and \`CL\` to 4.
2. Load value from memory at \`DI\` into \`AL\`.
3. Output \`AL\` to port C0.
4. Delay by looping until \`DX\` decrements to 0.
5. Increment \`DI\` and repeat until \`CL\` decrements to 0.
6. Loop back to step 1.

Code Converter

(A) ASCII to BCD

1. Load ASCII value into register AL.
2. Compare AL with '0' (30h).
3. If less than '0', store 11.
4. If between '0' and '9', convert by subtracting 30h.
5. Store the BCD value.

(B) BCD to ASCII

1. Load BCD value into register AL.
2. Compare AL with 0A.
3. If greater than or equal to 0A, store 11.
4. If less than 0A, convert by adding 30h.
5. Store the ASCII value.

(C) HEX to ASCII

1. Load HEX value into register AL.
2. Compare AL with 10h.
3. If less than 10h, check if less than 0Ah.
4. If less than 0Ah, convert by adding 30h; else, add 37h.
5. Store the ASCII value.

(D) ASCII to HEX

1. Load ASCII value into register AL.
2. Compare AL with '0' (30h).
3. If less than or equal to '9', convert by subtracting 30h.
4. If between 'A' and 'F', convert by subtracting 37h.
5. Store the HEX value.

Masm

MASM Program to Perform Addition

1. Prompt for the first number.
2. Read the first number.
3. Prompt for the second number.
4. Read the second number.
5. Add the two numbers.
6. Adjust for BCD.
7. Display the sum.

MASM Program to Perform Subtraction

1. Prompt for the first number.
2. Read the first number.
3. Prompt for the second number.
4. Read the second number.
5. Subtract the second number from the first.
6. Adjust for BCD.
7. Display the difference.

Algorithm to Display a String

1. Initialize the data segment.
2. Load the string address into DX.
3. Set AH to `09h`.
4. Call `int 21h` to display the string.
5. Exit the program with `int 21h`.

These steps summarize the main logic of each program without going into detail.

8051 logical operations:

(A) AND OPERATION

1. Take 1st number into register B.
2. Take 2nd number into register A.
3. Perform AND operation between registers A and B.
4. Store the result.

(B) OR OPERATION

1. Take 1st number into register B.
2. Take 2nd number into register A.

3. Perform OR operation between registers A and B.
4. Store the result.

(C) XOR OPERATION

1. Take 1st number into register B.
2. Take 2nd number into register A.
3. Perform XOR operation between registers A and B.
4. Store the result.

8051 bit arithmetic:

(A) ADDITION

1. Take 1st number into register B.
2. Take 2nd number into register A.
3. Add registers A and B.
4. Store the result.

(B) SUBTRACTION

1. Take 1st number into register B.
2. Take 2nd number into register A.
3. Subtract register B from A.
4. Store the result.

(C) MULTIPLICATION

1. Take 1st number into register B.
2. Take 2nd number into register A.
3. Multiply registers A and B.
4. Store the result.

(D) DIVISION

1. Take 1st number into register B.
2. Take 2nd number into register A.
3. Divide register A by B.
4. Store the quotient and remainder.