Computer Programming-1

Given a problem, shall we start coding immediately?

<u>Step 1</u>. What is the problem to be solved?

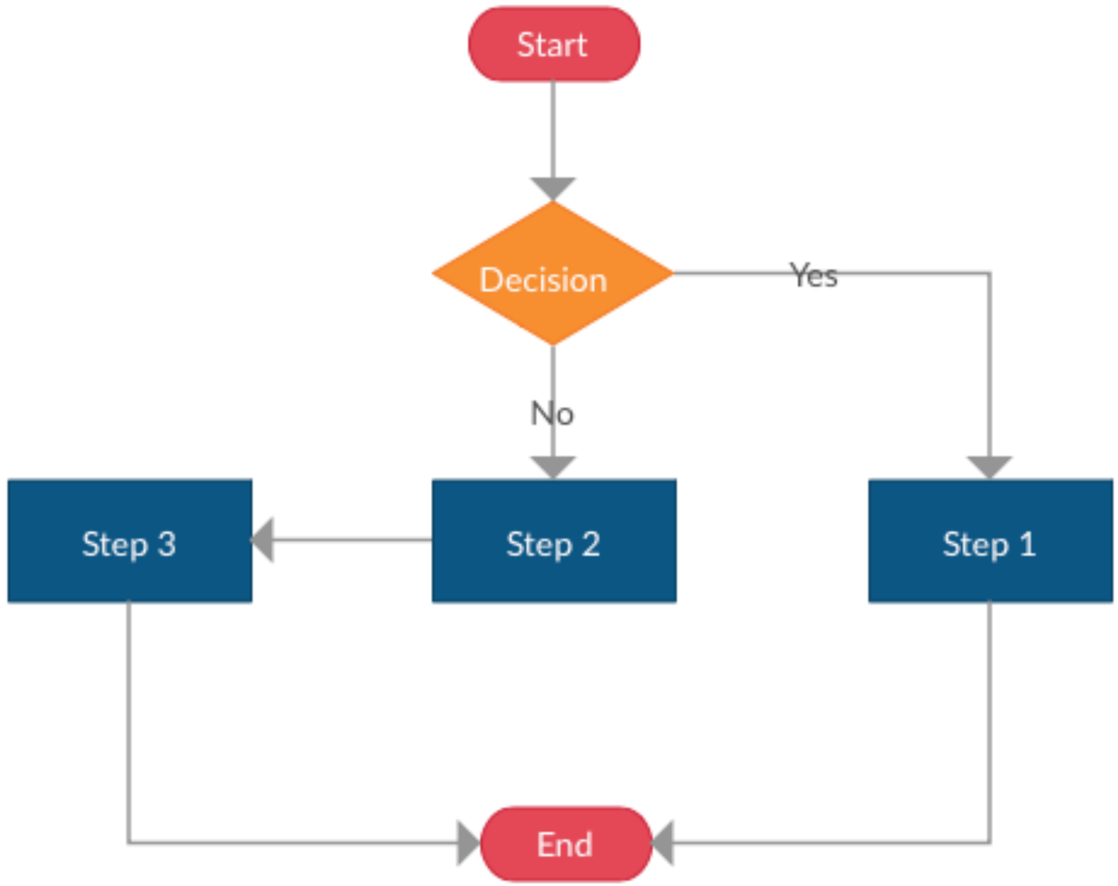
- Must have a good understanding of the situation
  - Then formulate Problem Statement
- Or from
  - Specification (given)

Computer Programming-2

Step 2. How to solve the problem?

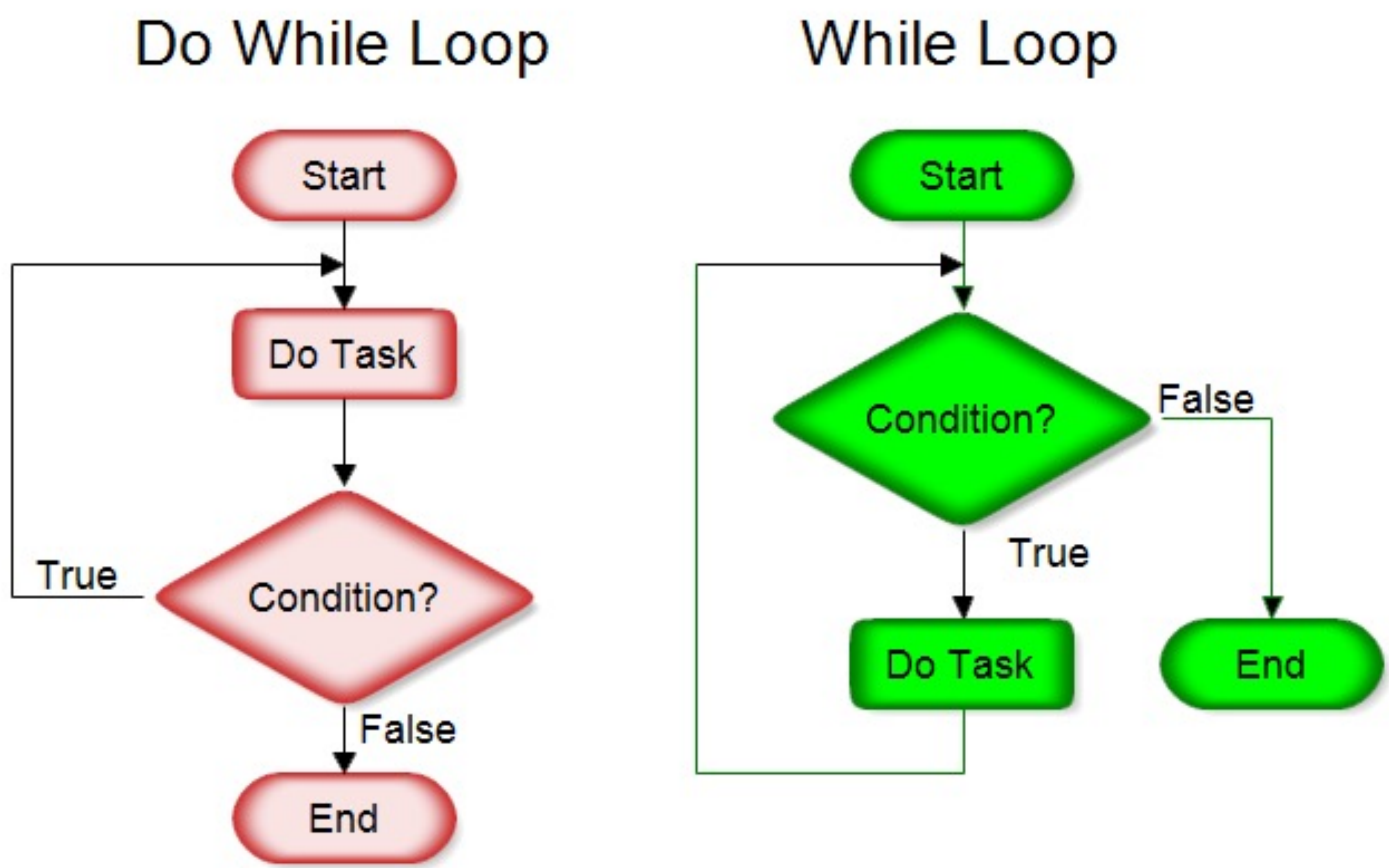
- Consider multiple approaches
- Use visual aids
  - Flowchart
  - Algorithm

Flow Chart: 2 paths & 1 decision



https://www.creaely.com/

Do Loop Flow Chart



https://www.rff.com/

Algorithm: Find Max from List

Specification: Given Input and Expected Output

Inputs: A list `L` of positive numbers. This list must contain at least one number. (Asking for the largest number in a list of no numbers is not a meaningful question.)

Outputs: A number `n`, which will be the largest number of the list.

Algorithm:

Algorithm: Input and Output Relationship

1. Set `max` to 0.
2. For each number `x` in the list `L`, compare it to `max`. If `x` is larger, set `max` to `x`.
3. `max` is now set to the largest number in the list.

http://www.fiftyexamples.readthedocs.io/

Computer Programming-3

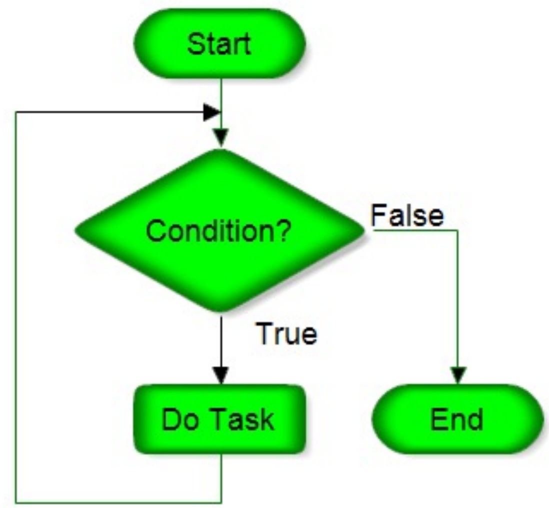
Step 3. Iterative Process:

- Idea
  - → Flowchart/Algorithm
    - → Coding
      - → Verification/Validation

Assembly Programming

1. Must have full understanding of the problem
   - From specification or problem formulation
2. Consider multiple approaches:
   - Use visual aids
     - Flowchart
     - Algorithm

While Loop



1. Set `max` to 0.
2. For each number `x` in the list `L`, compare it to `max`. If `x` is larger, set `max` to `x`.
3. `max` is now set to the largest number in the list.

Declare Data Structures

3. Define and allocate memory space:

- Variables
- Constants
- Arrays, Queues, Stacks

Problem Data Structures

- Counters
- Pointers
- Temporary Buffers / Storage

Program Data Structures

4. Write Pseudo Code

- Map flow chart or algorithm to pseudo code
- Visualize memory layout
- Focus on **logic**

```
        LD      R2, N
        CLR     R3
        MOV     R4, #NUM1
LOOP:   LD      R5, (R4)
        ADD     R3, R3, R5
        ADD     R4, R4, #4
        SUB     R2, R2, #1
        BGT     R2, R0, LOOP
        ST      R3, SUM
```

```
SUM:    RESERVE     4
N:      DATAWORD    150
NUM1:   RESERVE     600
```

Assignment Pseudo to Thumb

- Convert these pseudo codes into Thumb instructions

```
        LD      R2, N
        CLR     R3
        MOV     R4, #NUM1
LOOP:   LD      R5, (R4)
        ADD     R3, R3, R5
        ADD     R4, R4, #4
        SUB     R2, R2, #1
        BGT     R2, R0, LOOP
        ST      R3, SUM

SUM:    RESERVE     4
N:      DATAWORD    150
NUM1:   RESERVE     600
```

5. Coding Using ISA Manual

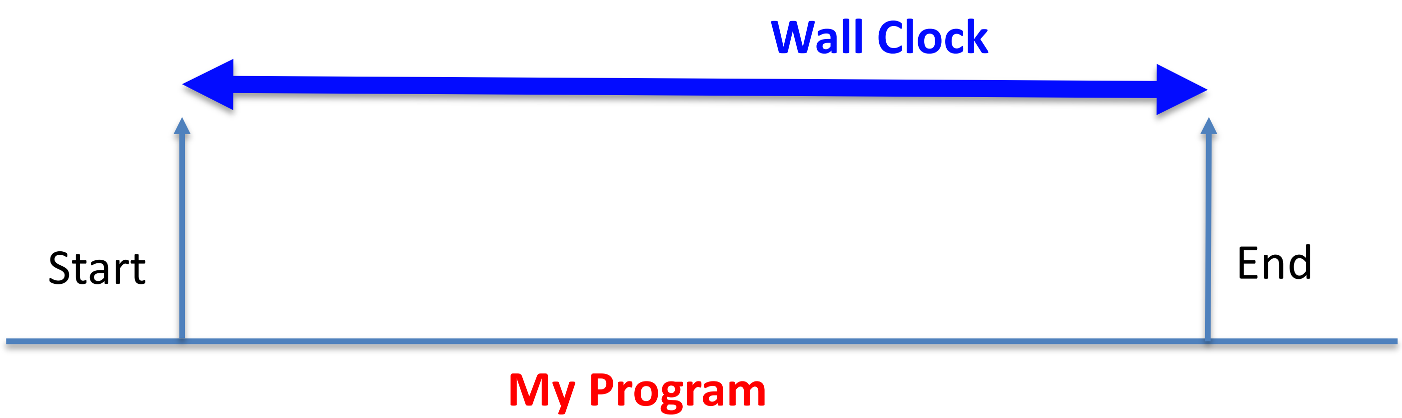
- Write Program / Subroutines (based on pseudo code)
- Focus on **syntax and semantics**
  - Initializations
    - Counters; Pointers, etc.
  - Load / Store Architecture
  - Operand Locations
  - Control Structures

Control Structures

- Exercise: Lab #2
    1. If *T* then *B*; else *C*
    2. Switch (x=variable)
        a) case *0*:…; case *1*:…; …; case *x*:…
        b) If x=0, then do; if x=1, then do; if x=2, then do; if x=3, then do
    3. While *T* do …; Do… while *T*
    4. For *i* = 0..*n*, do …
- Based on Branch CC (N, Z, C, V)

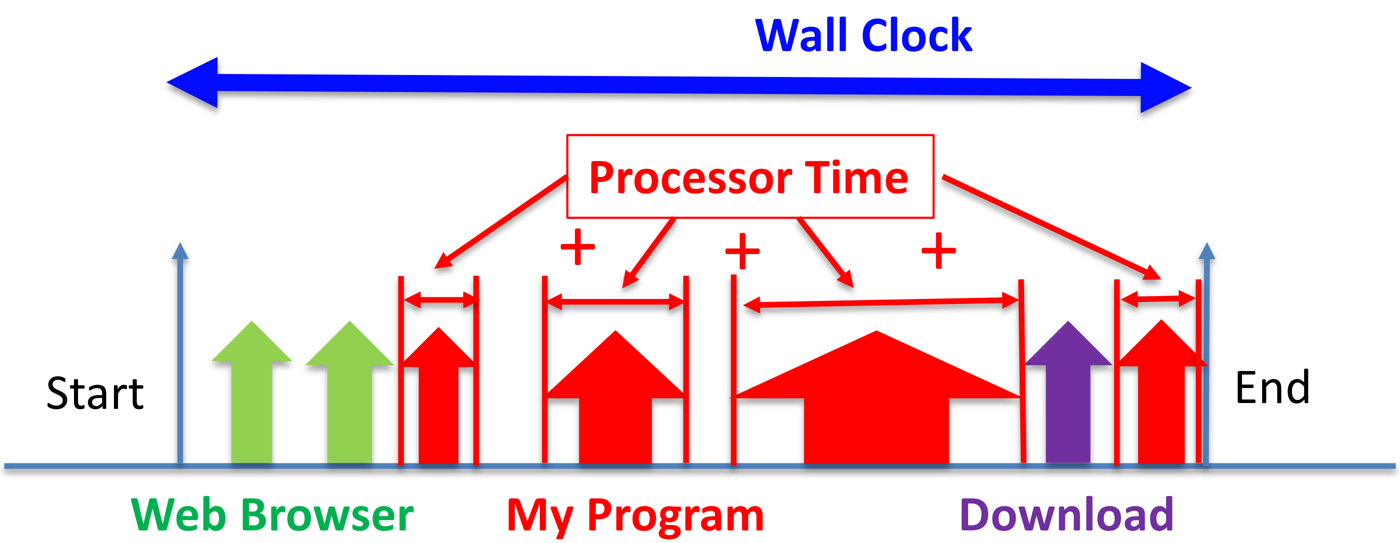
Program Execution Time

- **Elapsed time:** the wall clock time to execute a program



Processor Execution Time

- **Processor time:** the time periods that the processor is executing **MY** program

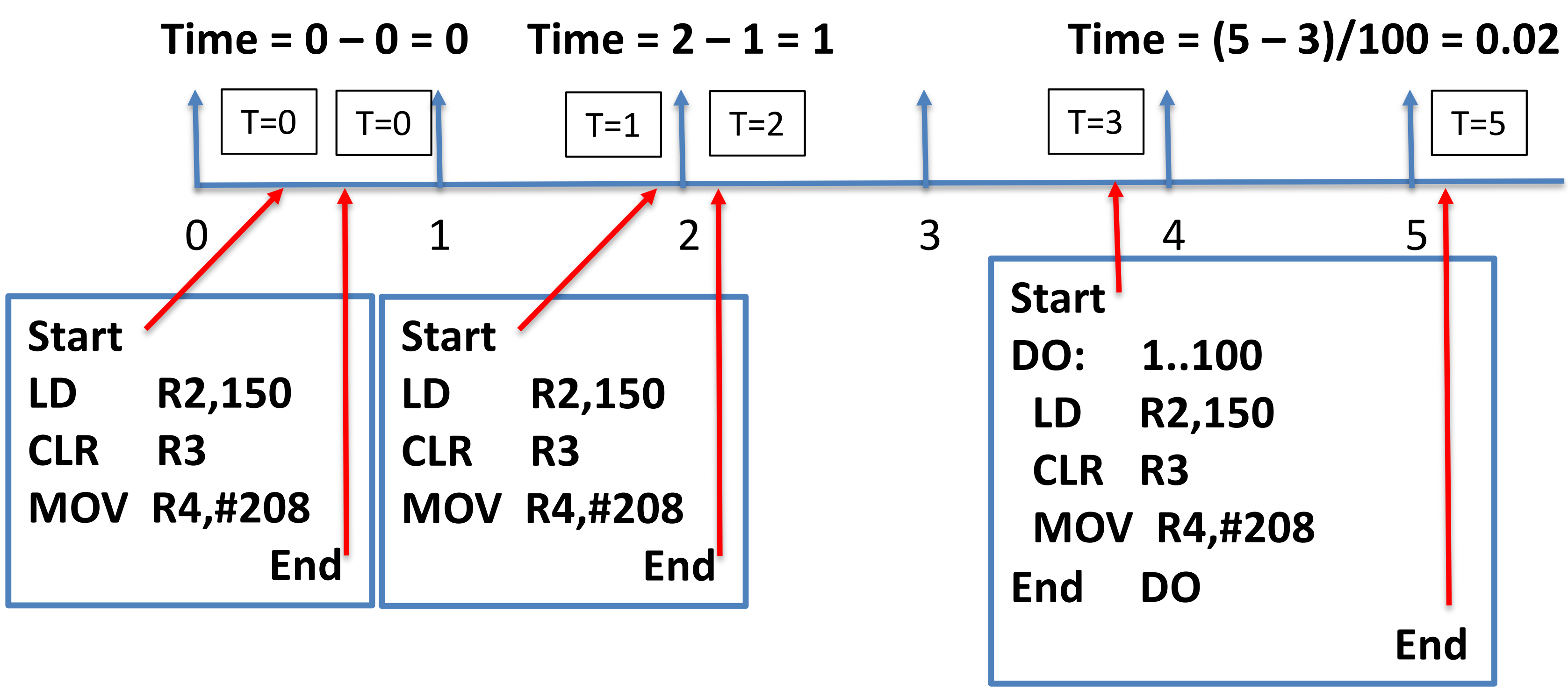


Time Measurement

- Resolution of the timer: Timer Granularity
- Example: 1 time unit resolution (1 tick)

What is time granularity?

The size of time interval!



Time Measurement Functions

- *C*: CPU time
  - Start_time = clock (); work…; End_time = clock ()
  - Time = End_time – Start_time
- *Java*: Wall Clock time
  - System.currentTimeMillis()
- *Unix*: $time program.o
  - wall-clock, user-cpu, system-cpu

```
Start_time = clock()
    Inst-1
    Inst-2
    …
    …
    …
    Inst-n
End_time = clock()
```