Introduction to Computing Systems
from bits & gates to C & beyond

Chapter 6

Programming

- *Problem solving*
- *Debugging*

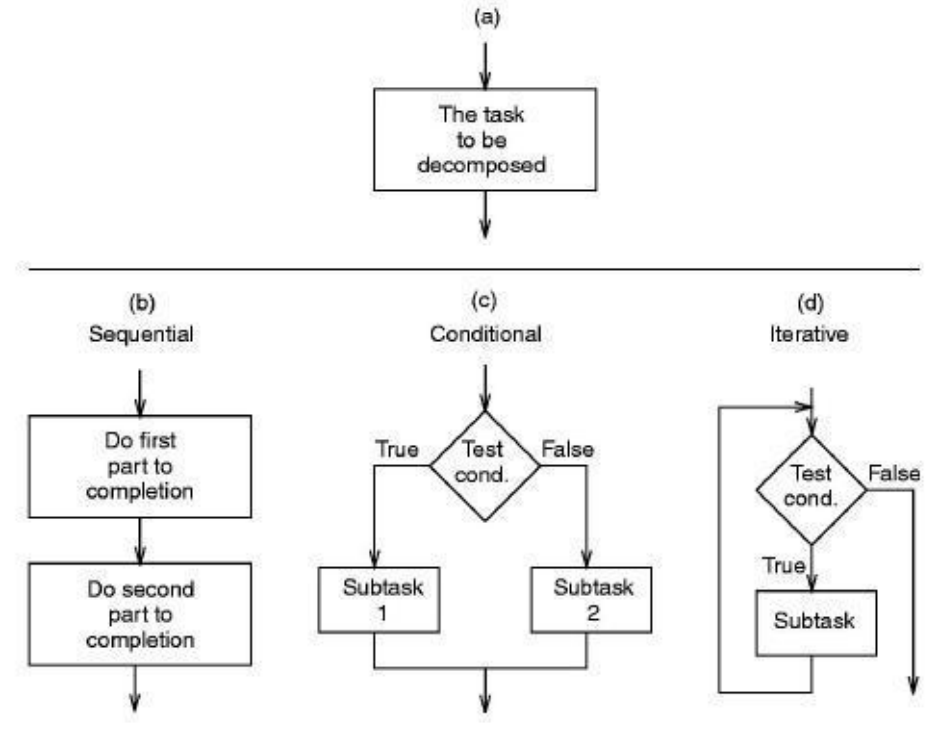
Problem solving

- **Start with systematic decomposition of problem**
  - **“top-down” analysis**
  - **stepwise refinement**
- **Algorithms must have properties of:**
  - **finiteness**
  - **completeness**
  - **definiteness**
  - **computability**

Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

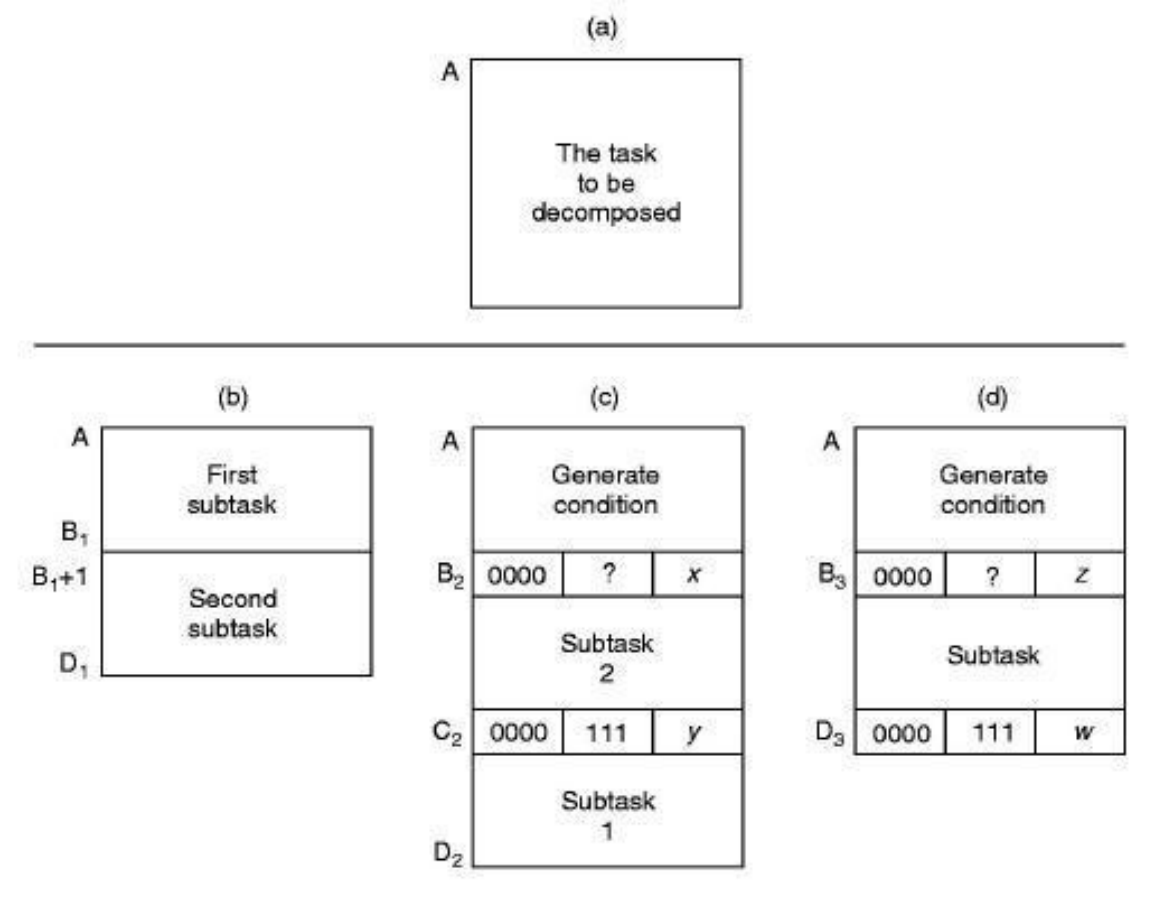
Structured programming

- **Three control structures:**
  - **Sequential**
    - ***This is the default***
  - **Conditional**
    - ***Branching or decision-making***
  - **Iteration**
    - ***Loops***



Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

Implementing the control structures

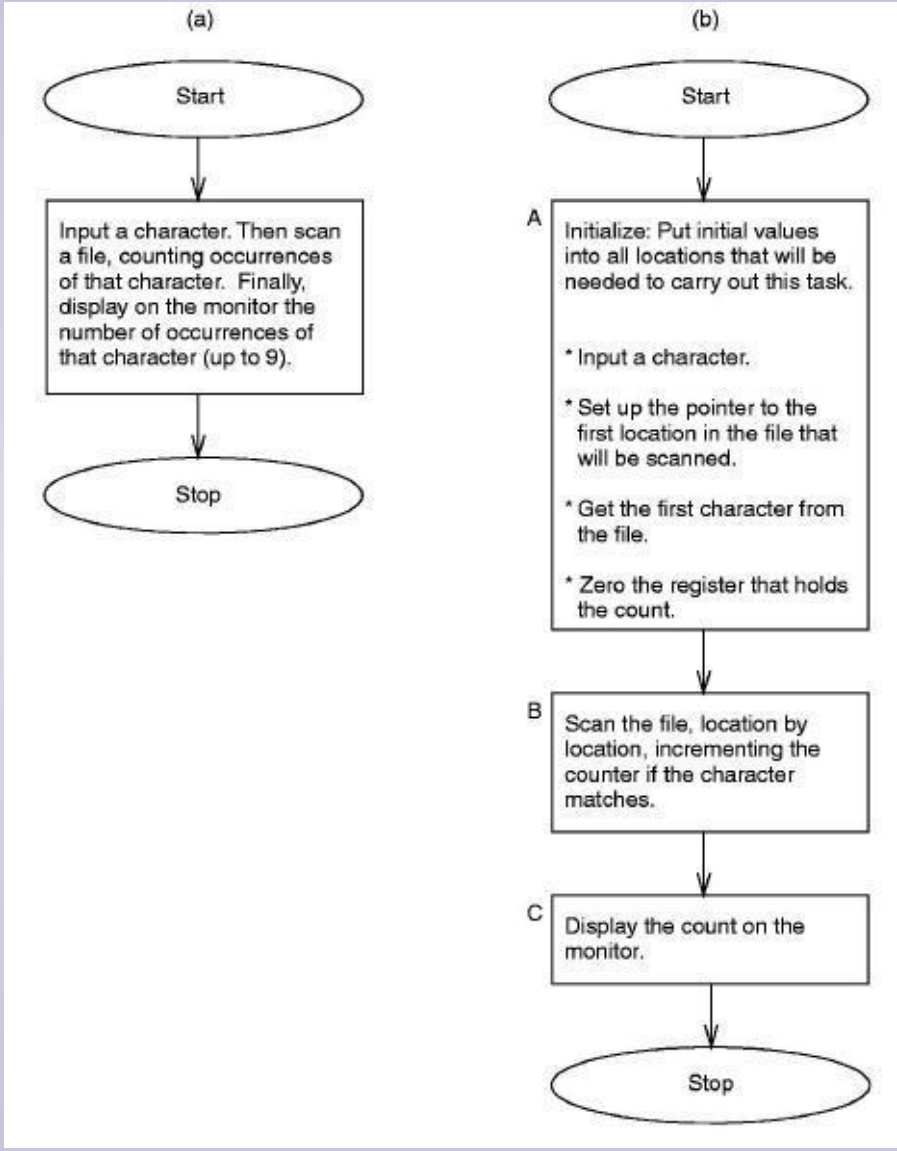


Sequential **Conditional** **Iterative**

Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

Stepwise refinement illustrated

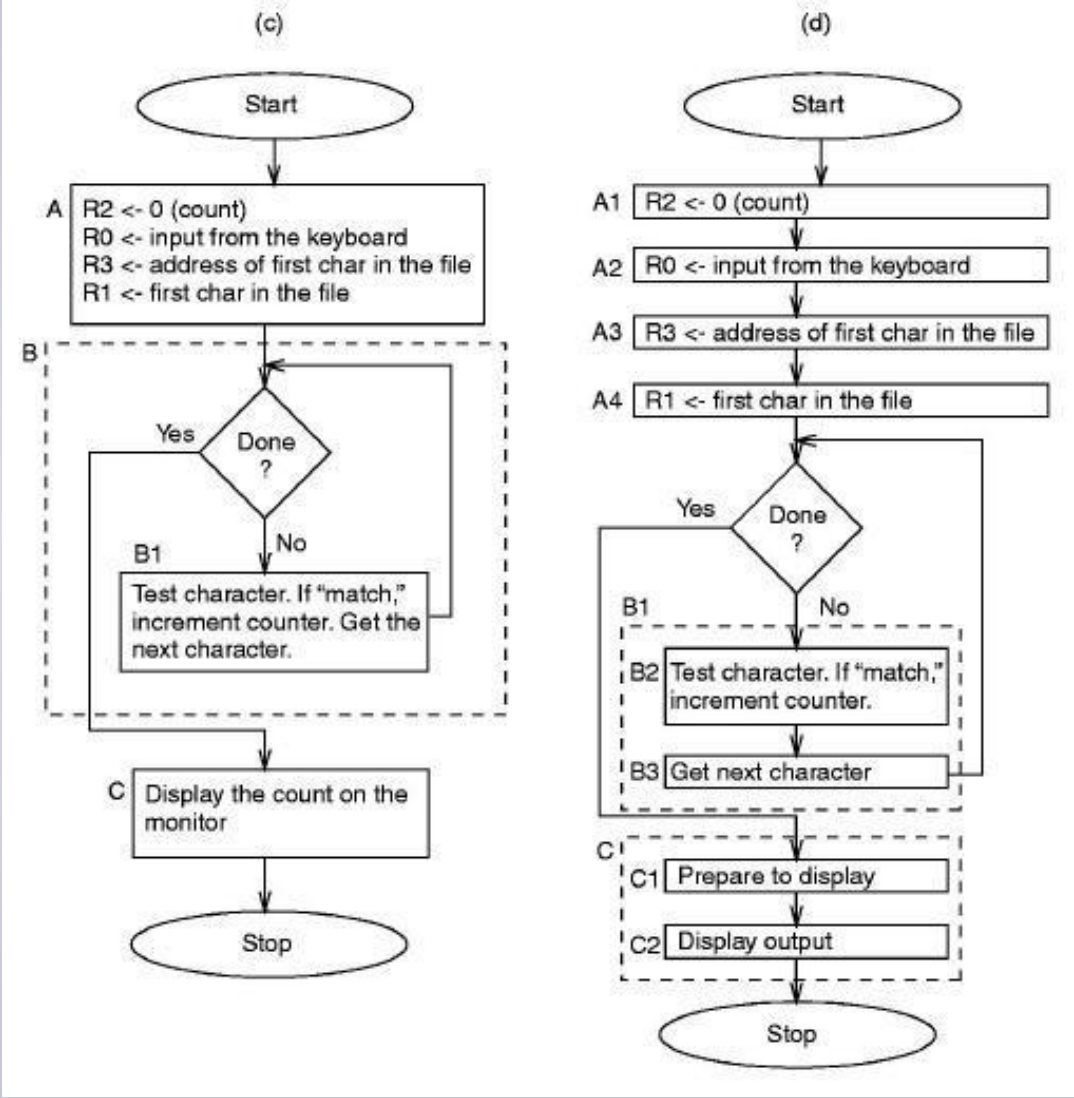
- **Character count algorithm:**
  - **statement of problem & first level of refinement**



Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

Stepwise refinement (cont.)

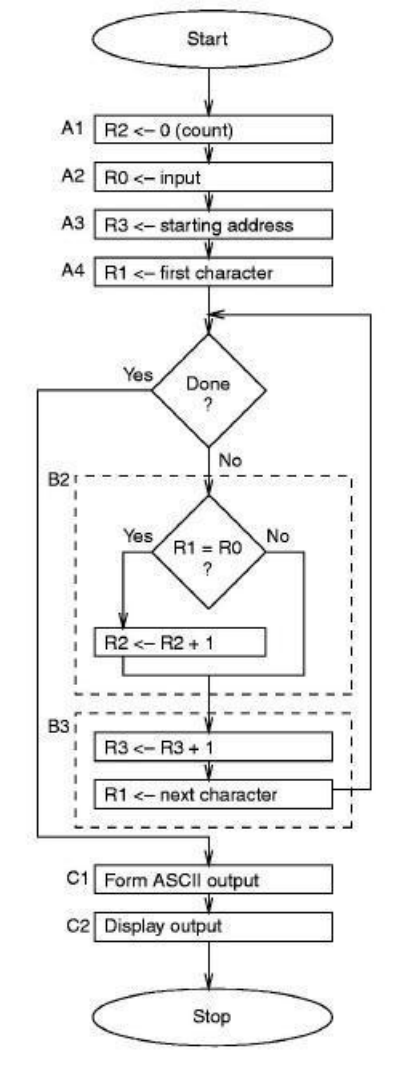
- 2nd & 3rd levels



Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

Stepwise refinement (cont.)

- **Final level**



Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

Debugging

- **A debugging tool provides (at least) the ability to:**
  - **Set values to registers & memory locations**
    - ***enter as hex, decimal or binary values***
  - **Execute instructions one at a time, or in small groups**
    - ***Run will set the program running***
    - ***Step causes a fixed number of instructions to be executed (often used to "single-step" through part of a program)***
  - **Stop execution when desired**
    - ***Breakpoints allow the user to set points at which execution will halt & wait for a new "Run" instruction***
  - **Examine the contents of registers and memory locations at any point**

Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

Debugging example

- **A program to find the first 1 in a 16-bit word**

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
x3000	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	0	`R1 <- 0`
x3001	0	0	0	1	0	0	1	0	0	1	1	0	1	1	1	1	`R1 <- R1 + 15`
x3002	1	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	`R2 <- M[M[x3009]]`
x3003	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	`BRn x3008`
x3004	0	0	0	1	0	0	1	0	0	1	1	1	1	1	1	1	`R1 <- R1 - 1`
x3005	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	`R2 <- R2 + R2`
x3006	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	`BRn x3008`
x3007	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	`BRnzp x3004`
x3008	1	1	1	1	0	0	0	0	0	0	1	0	0	1	0	1	`HALT`
x3009	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	`x3100`

Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside

Debugging example (cont.)

Oops! We have an infinite loop!

Setting a breakpopint at location x3007 & repeatedly running the code shows that the loop does not trerminate because there is no 1 in the test case number

PC	R1
x3007	14
x3007	13
x3007	12
x3007	11
x3007	10
x3007	9
x3007	8
x3007	7
x3007	6
x3007	5
x3007	4
x3007	3
x3007	2
x3007	1
x3007	0
x3007	-1
x3007	-2
x3007	-3
x3007	-4

Slides prepared by Walid A. Najjar & Brian J. Linard, University of California, Riverside