University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Systematic Decomposition

How Does One Write a Program?

You have seen several examples of programming.

Given a task in human terms,

- we **produce an algorithm**
- that solves the problem
- using steps that each require a few **LC-3** instructions (or **C** statements).

How did we do it?

Systematic Decomposition: What is It?

Systematic decomposition is an approach to programming. The idea is as follows:

- **starting with a high-level model** of the task, usually in a human language,
- repeatedly **break** the task **into simpler tasks**
- until **each subtask** is easily **expressed in a few instructions**.

We Will Discuss Three Constructs

We will discuss

- the pieces (the structure of "simpler tasks")
- and how each maps to **LC-3** memory.

But before we start, a couple of comments on programming…

Don’t Underestimate the Value of Having a Model

Pencil and paper are your first tools.

If your algorithm is clear in your head,

- when your code has bugs,
- you will find it easier to spot the differences
- between what you meant to write
- and what you wrote.

Draw pictures, draw flow charts, think.

Then sit down to write the code.

© 2016 Steven S. Lumetta. All rights reserved.

Write Comments First

When you do get ready to write your program:

First, write comments that describe tasks at intermediate levels.

Then fill in the code for each comment.

Don’t leave comments as an afterthought.

© 2016 Steven S. Lumetta. All rights reserved.

Break Down Tasks Using One of Three Constructs

What do “simpler tasks” look like?

Typically, they form one of three patterns.

You have seen these patterns before:

- they correspond to statements in **C**,
- but the iterative construct is simpler.

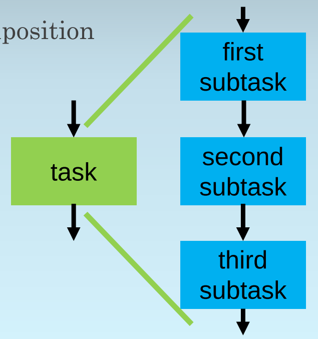
Let’s take a look.

© 2016 Steven S. Lumetta. All rights reserved.

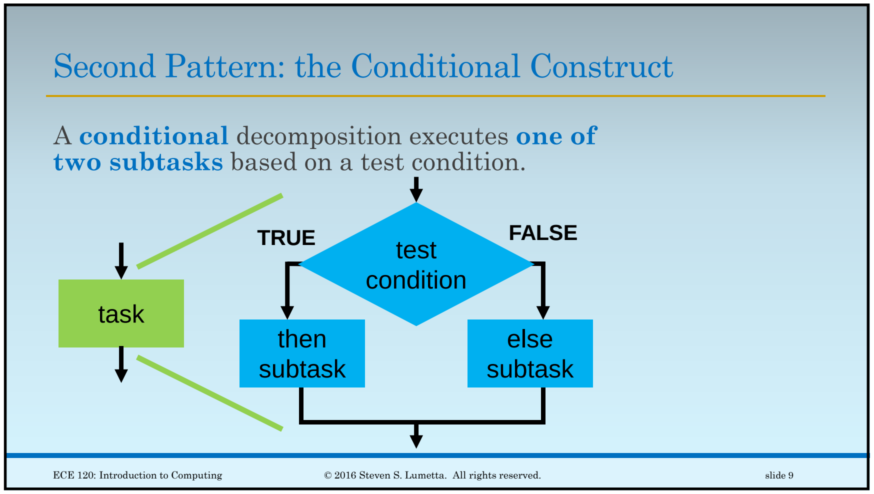
First Pattern: the Sequential Construct

A **sequential** decomposition breaks the task

- into **two or more subtasks**
- executed **in sequence**.



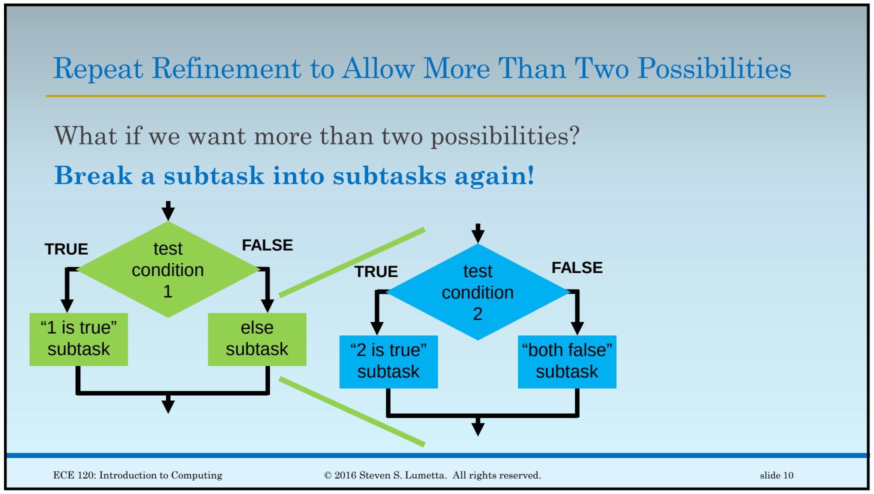
© 2016 Steven S. Lumetta. All rights reserved.



Second Pattern: the Conditional Construct

A **conditional** decomposition executes **one of two subtasks** based on a test condition.

© 2016 Steven S. Lumetta. All rights reserved.

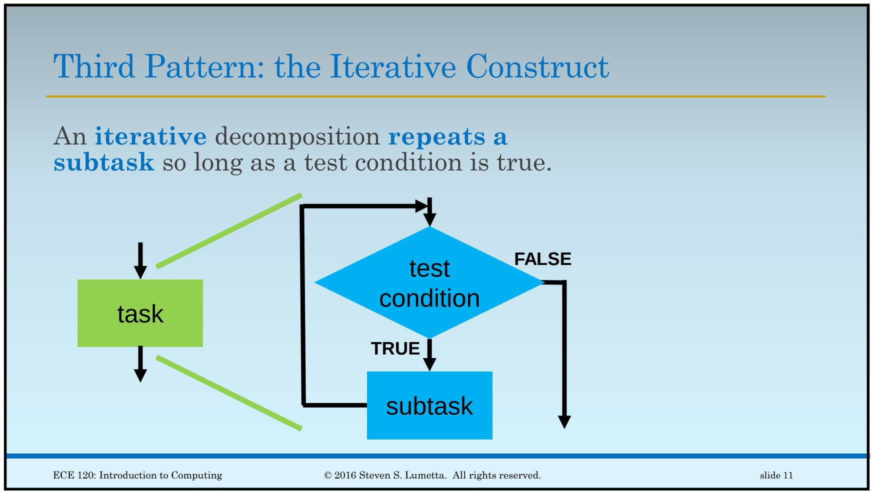


Repeat Refinement to Allow More Than Two Possibilities

What if we want more than two possibilities?

Break a subtask into subtasks again!

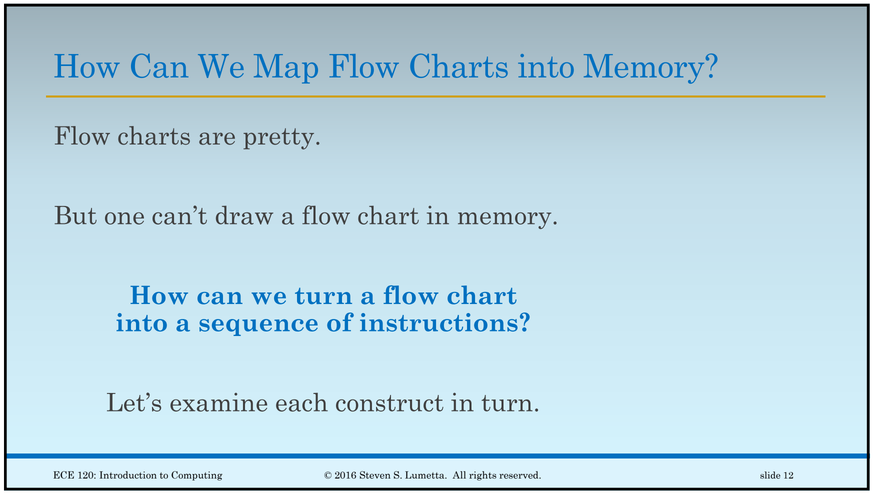
© 2016 Steven S. Lumetta. All rights reserved.



Third Pattern: the Iterative Construct

An **iterative** decomposition **repeats a subtask** so long as a test condition is true.

© 2016 Steven S. Lumetta. All rights reserved.



How Can We Map Flow Charts into Memory?

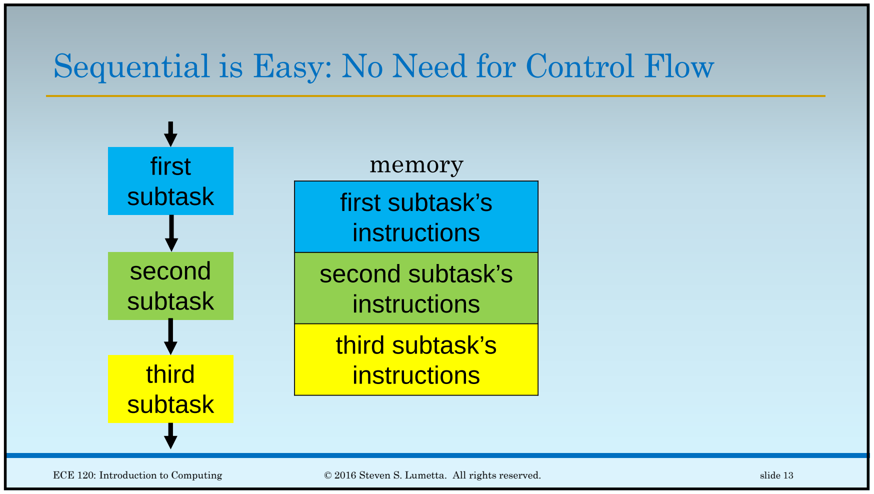
Flow charts are pretty.

But one can't draw a flow chart in memory.

How can we turn a flow chart into a sequence of instructions?

Let's examine each construct in turn.

© 2016 Steven S. Lumetta. All rights reserved.



Sequential is Easy: No Need for Control Flow

first subtask

second subtask

third subtask

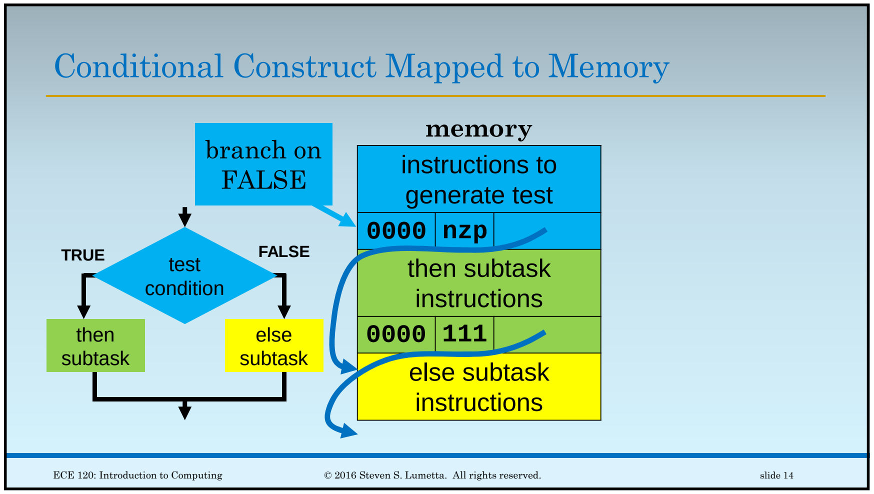
memory

first subtask's instructions

second subtask's instructions

third subtask's instructions

© 2016 Steven S. Lumetta. All rights reserved.



Conditional Construct Mapped to Memory

branch on FALSE

TRUE

test condition

FALSE

then subtask

else subtask

memory

instructions to generate test

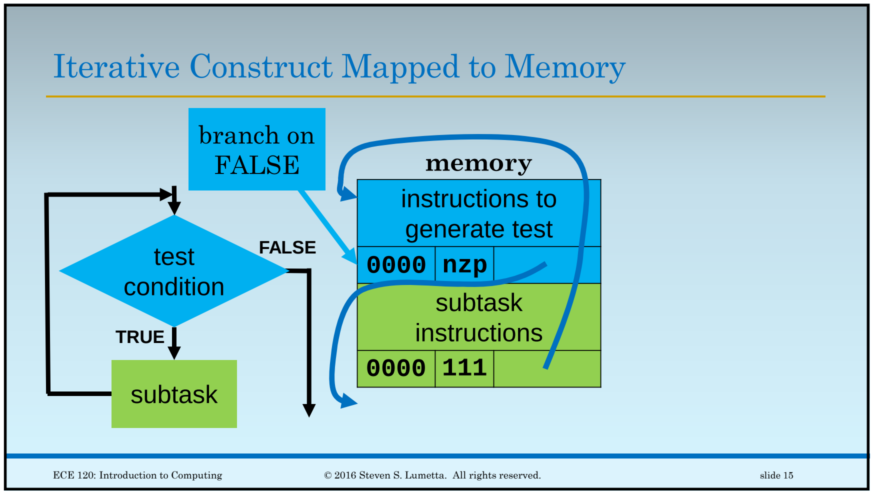
0000 nzp

then subtask instructions

0000 111

else subtask instructions

© 2016 Steven S. Lumetta. All rights reserved.



Iterative Construct Mapped to Memory

branch on FALSE

test condition

FALSE

TRUE

subtask

memory

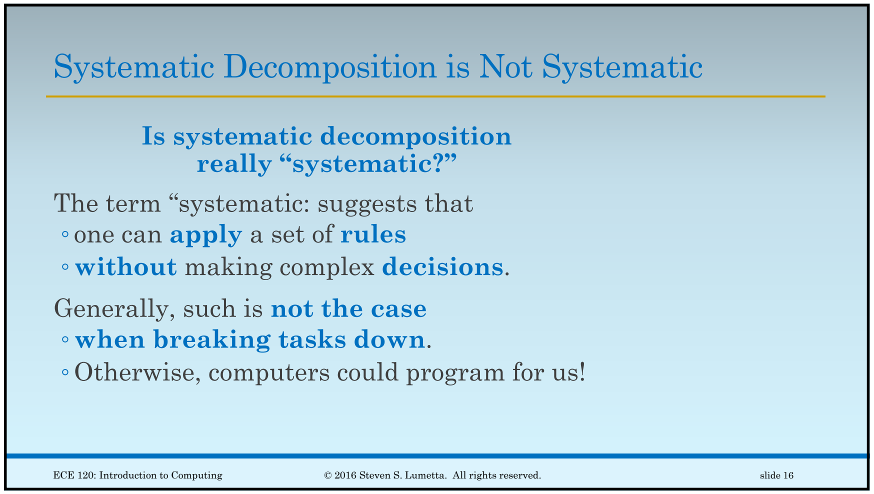
instructions to generate test

0000 nzp

subtask instructions

0000 111

© 2016 Steven S. Lumetta. All rights reserved.



Systematic Decomposition is Not Systematic

Is systematic decomposition really "systematic?"

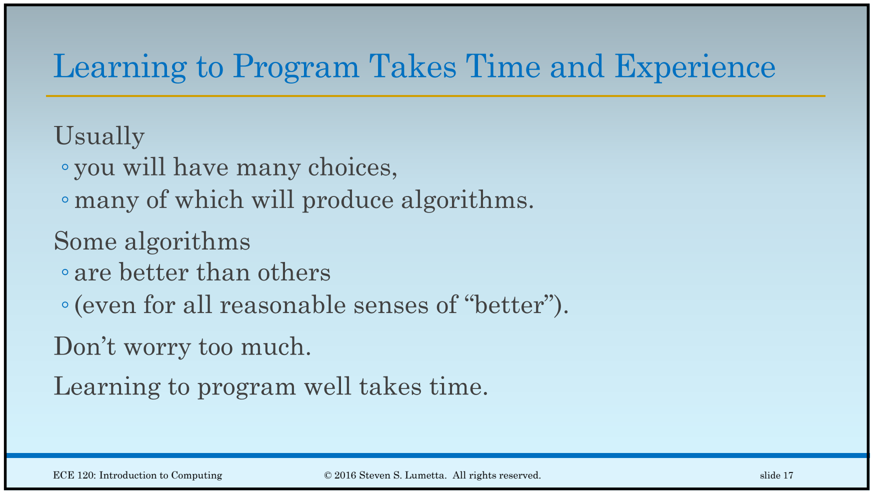
The term "systematic: suggests that

- one can **apply** a set of **rules**
- **without** making complex **decisions**.

Generally, such is **not the case**

- **when breaking tasks down**.
- Otherwise, computers could program for us!

© 2016 Steven S. Lumetta. All rights reserved.



Learning to Program Takes Time and Experience

Usually

- you will have many choices,
- many of which will produce algorithms.

Some algorithms

- are better than others
- (even for all reasonable senses of “better”).

Don’t worry too much.

Learning to program well takes time.

© 2016 Steven S. Lumetta. All rights reserved.