



CprE 281: Digital Logic

Instructor: Alexander Stoytchev

<http://www.ece.iastate.edu/~alexs/classes/>

Register Machines

CprE 281: Digital Logic
Iowa State University, Ames, IA
Copyright © Alexander Stoytchev

Administrative Stuff

- **Homework 12 is due on Wednesday @ 10 pm**

Administrative Stuff

- **Extra Credit Lab is due on Friday Dec 2 @ 10 pm**

Administrative Stuff

- **Final Project: due during your last lab (dead week).**
 - **The grading rubric is posted on the class web page.**
 - **presentation of your work to your TAs**
 - **final report pdf (upload on canvas)**
 - **zip file with your design (upload on canvas)**
 - **If you work in a team: each student needs to present separately to their lab TAs.**

Administrative Stuff

- The FINAL exam is scheduled for

Wednesday Dec 14 @ 2:15 – 4:15 PM

Final Exam Format

- **The exam will cover: Chapter 1 to Chapter 6, and Sections 7.1-7.2, register machines, and i281 CPU**
- **Emphasis will be on Chapter 5, 6, and 7**
- **The exam will be closed book but open notes.**
- **You can bring up to 5 pages of handwritten or typed notes.**

Final Exam Format

- **The exam will be out of 135 points**
- **You need 95 points to get an A on this exam**
- **It will be great if you can score more than 100 points.**
 - **but you can't roll over your extra points 😞**

Topics for the Final Exam

- **K-maps for 2, 3, and 4 variables**
- **Multiplexers (circuits and function)**
- **Synthesis of logic functions using multiplexers**
- **Shannon's Expansion Theorem**
- **1's complement and 2's complement representation**
- **Addition and subtraction of binary numbers**
- **Circuits for adding and subtracting**
- **Serial adder**
- **Latches (circuits, behavior, timing diagrams)**
- **Flip-Flops (circuits, behavior, timing diagrams)**
- **Counters (up, down, synchronous, asynchronous)**
- **Registers and Register Files**

Topics for the Final Exam

- **Synchronous Sequential Circuits**
- **FSMs**
- **Moore Machines**
- **Mealy Machines**
- **State diagrams, state tables, state-assigned tables**
- **State minimization**
- **Designing a counter**
- **Arbiter Circuits**
- **Reverse engineering a circuit**
- **ASM Charts**
- **Register Machines and programs for them**
- **ALU, PC, and control for a simple processor (i281 CPU)**
- **Assembly and machine language (i281 assembly)**
- **Something from Star Wars**

Reading Material for Today

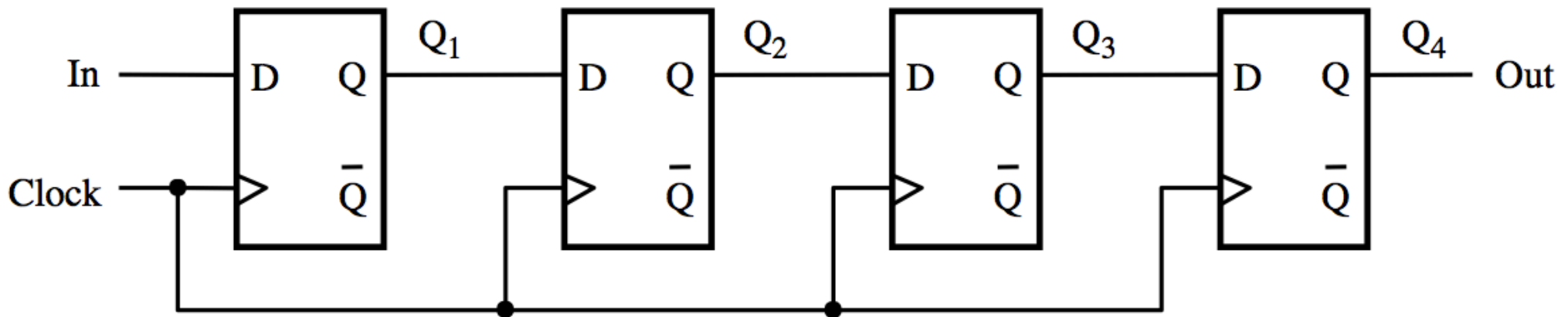
- **“The Seven Secrets of Computer Power Revealed” by Daniel Dennett.**
- **This is Chapter 24 in his latest book “Intuition Pumps and Other Tools for Thinking”, 2013**

Reading Material for Today

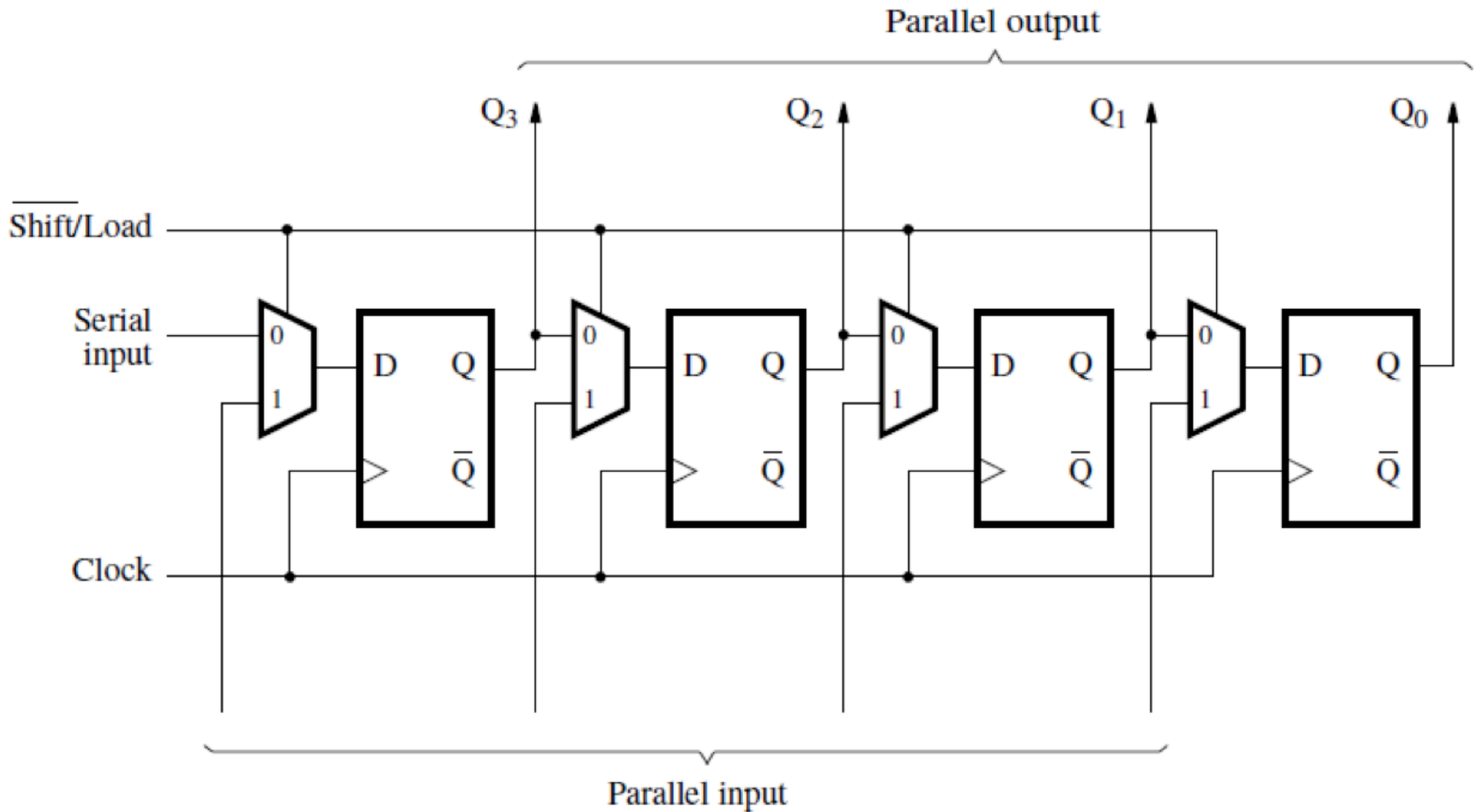
- **“The Seven Secrets of Computer Power Revealed” by Daniel Dennett.**
- **This is Chapter 24 in his latest book “Intuition Pumps and Other Tools for Thinking”, 2013**
- **Chapter 24 in reverse is: 42!**

What is a register?

What is a register?

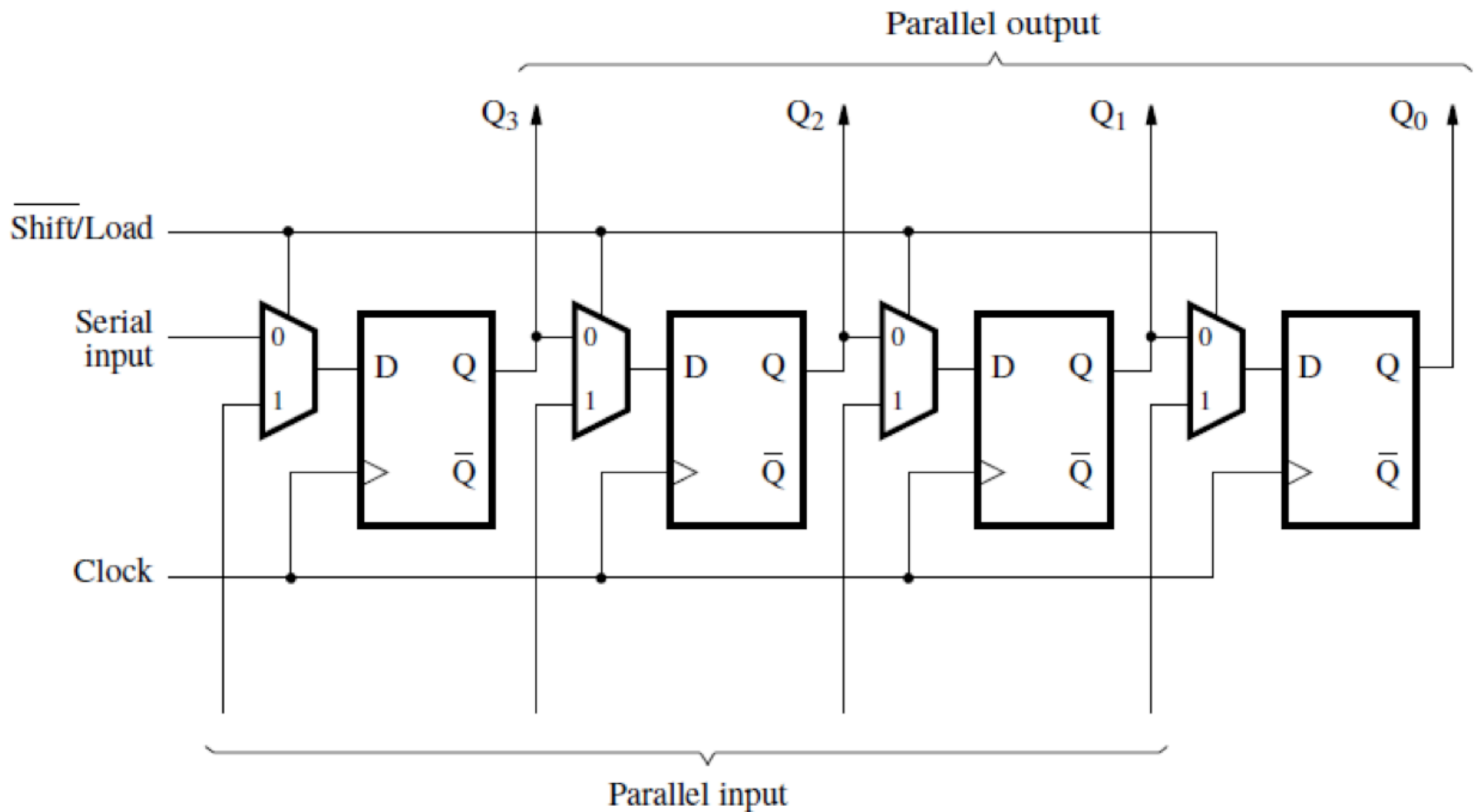


What is a register?



[Figure 5.18 from the textbook]

What can be stored in this register?



We Need a Simpler Abstraction

What is a register?

What is a register?



What can be stored in this register?



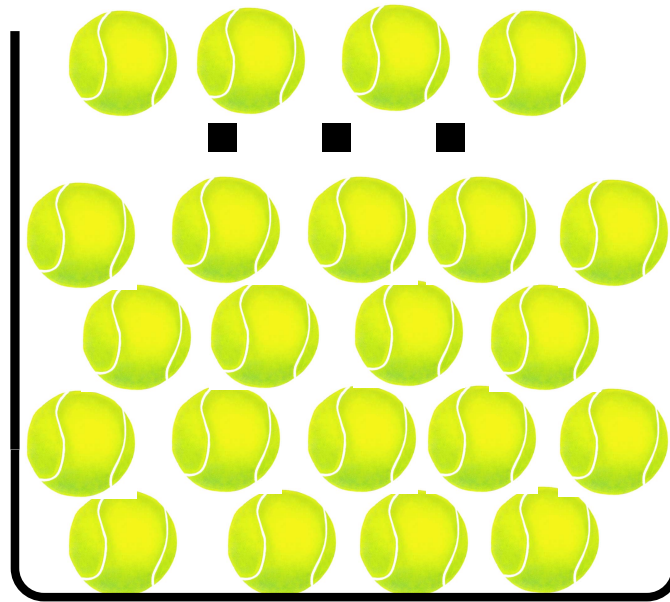
What can be stored in this register?



How many balls can be stored?



How many balls can be stored?



Infinitely Many

What is a register machine?

What is a register machine?



What is a register machine?



Register 1



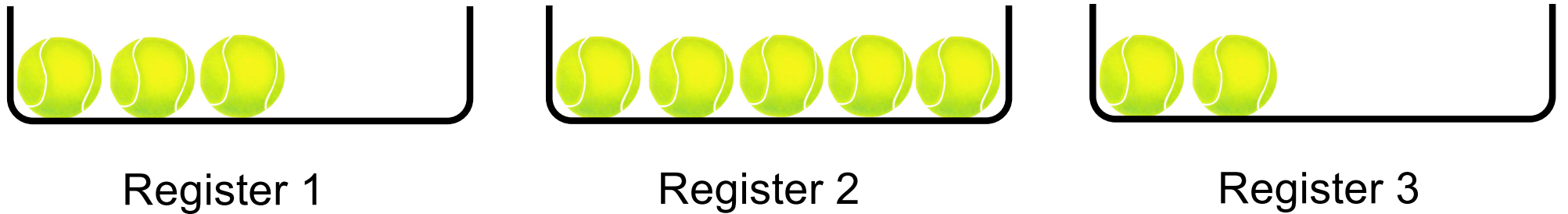
Register 2



Register 3

The registers are numbered

What is a register machine?



The registers are numbered and each could store a different number of balls.

Things to Notice

- **There could be many registers, not just three, as long as they are finitely many.**
- **The address of each register is an integer.**
- **Each register stores an integer number of balls.**
- **Storing 3.14 balls in a register is not possible.**

**In addition to the registers
there is also a processing unit**

In addition to the registers there is also a processing unit

- **It operates on the registers by changing their contents**
- **The operations are ordered sequentially and given a number (just like line numbers in BASIC)**
- **There are only three possible operations**
 - **Increment a register and go to another step**
 - **Decrement a register and go to another step**
 - **End**

Two Basic Operations

(that can be performed on a register)

- **Increment the contents of a register by 1**
- **Decrement the contents of a register by 1**
- **In both cases the address of the register must be given**

Incrementing a Register



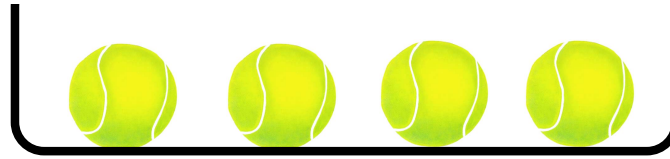
Register 5

Incrementing a Register



Register 5

Incrementing a Register



Register 5

Decrementing a Register



Register 3

Decrementing a Register



Register 3

Decrementing a Register



Register 3

Decrementing a Register



Register 3

Decrementing a Register

This is our if statement



Register 3

Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3

This register is not used by this program.

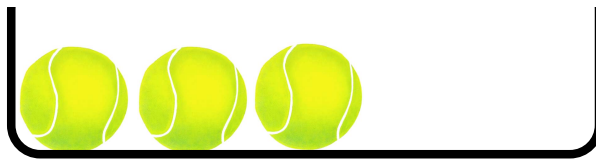
STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3



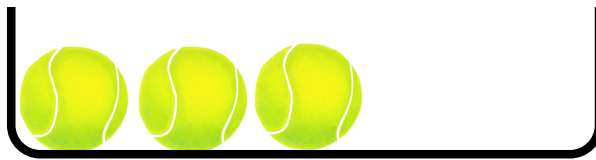
STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3



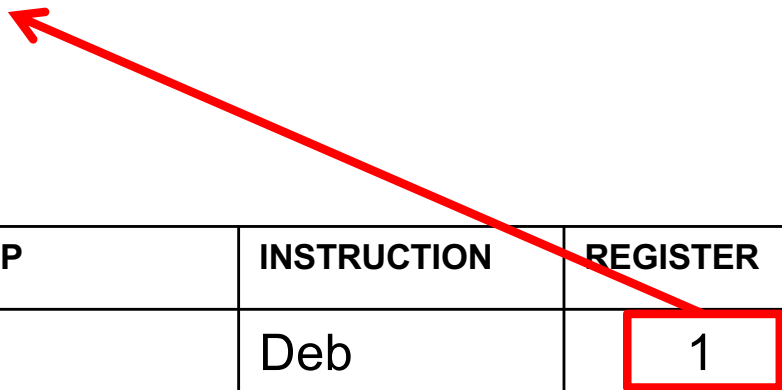
STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

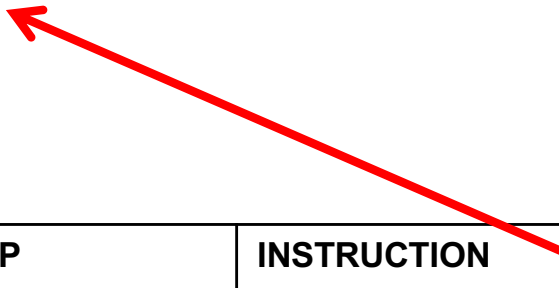


Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			



Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			



Sample Program:

Add the contents of register 1 to the contents of register 2



⇒

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

→

A red arrow points from the '2' in the 'REGISTER' column of step 2 up to Register 2 in the diagram above. A red box highlights the '2' in the 'REGISTER' column of step 2.

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

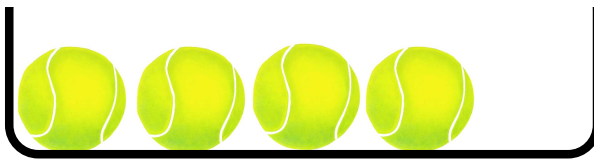
A red arrow points from the 'REGISTER' column of the second row (Step 2) to Register 2 in the diagram above. A red box highlights the number '2' in the 'REGISTER' column of the second row. A red arrow points to the first row of the table.

Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			



Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1. ←	Deb	1	2	3
⇒ 2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3



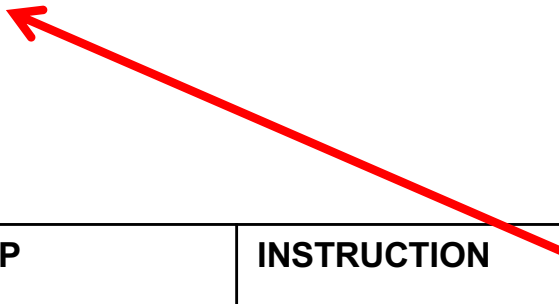
STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

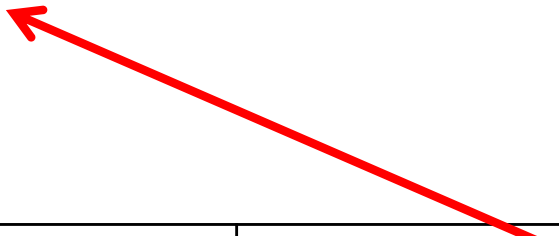


Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			



Sample Program:

Add the contents of register 1 to the contents of register 2



⇒

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

A red arrow points from the right side of the table to the left. A red arrow also points from the 'GO TO STEP' cell '2' in the first row to the 'INSTRUCTION' cell 'Inc' in the second row.

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			



Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

A red arrow points to the first column of the table, and a red box highlights the value '2' in the 'REGISTER' column of the second row.

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

→

A red arrow points from the '2' in the 'REGISTER' column of step 2 up to Register 2 in the diagram above. A red box highlights the '2' in the 'REGISTER' column of step 2.

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

A red arrow points from the 'REGISTER' column of step 2 to Register 2 in the diagram above. A red arrow also points to the 'REGISTER' column of step 2 from the left.

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

⇒

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

A red arrow points from the 'GO TO STEP' cell '1' in row 2 to the 'STEP' cell '1.' in row 1. A red arrow also points to the 'STEP' cell '2.' in row 2.

Sample Program:

Add the contents of register 1 to the contents of register 2



⇒

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



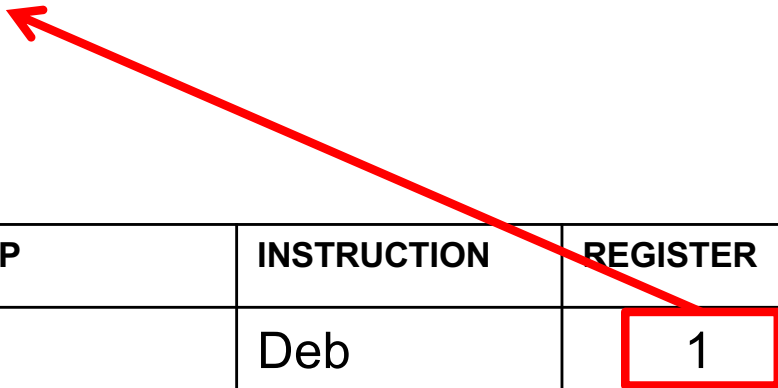
STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			



Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

A red arrow points from the right to the first row of the table. A red arrow points from the circled '3' in the first row to the '3.' in the third row.

Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			



Sample Program:

Add the contents of register 1 to the contents of register 2



STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Same program, different initial conditions

Add the contents of register 1 to the contents of register 2



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Some Questions

- **Where do the balls that are deleted go?**
- **Where do the balls that are added come from?**

Same program, different initial conditions

Add the contents of register 1 to the contents of register 2



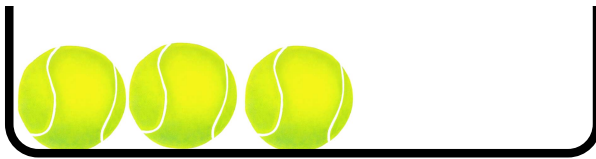
STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	1	2	3
2.	Inc	2	1	
3.	End			

Another Program:

Move the contents of register 2 to register 3



Register 1



Register 2



Register 3

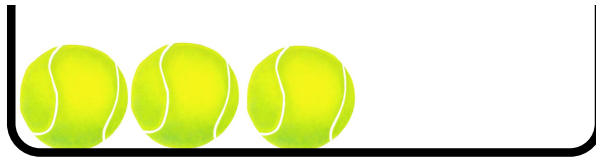
STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	3	1	2
2.	Deb	2	3	4
3.	Inc	3	2	
4.	End			

Yet Another Program:

Copy the contents of register 1 to register 3



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	3	1	2
2.	Deb	2	2	3
3.	Deb	1	4	6
4.	Inc	3	5	
5.	Inc	2	3	
6.	Deb	2	7	8
7.	Inc	1	6	
8.	End			

RodRego

You can download a register machine emulator from:

<http://sites.tufts.edu/rodrego/>

It works on both Macs and PCs. It's pretty cool.

This web link is also in the book chapter.

There is also a version that works in a browser:

<https://rodrego.it.tufts.edu/>

What are the seven secrets?

Secret #1

Competence Without Comprehension - the register machine can do perfect arithmetic without having to comprehend what it is doing.

Secret #2

What a number in a register stands for depends on the program that we have composed.

Secret #3

Since the number in a register machine can stand for anything, this means that the register machine can, in principle, be designed to “notice” anything, to “discriminate” any pattern or feature that can be associated with a number – or a number of numbers.

Secret #4

Since a number can stand for anything, a number can stand for an instruction or an address.

Secret #5

All possible programs can be given a unique number as a name, which can then be treated as a list of instructions to be executed by a Universal machine.

Secret #6

All the improvements in computers since Turing invented his imaginary paper-tape machine are simply ways of making them faster.

Secret #7

There are no more secrets.

What about real computers?

- Do they really work in this way?

Questions?

THE END