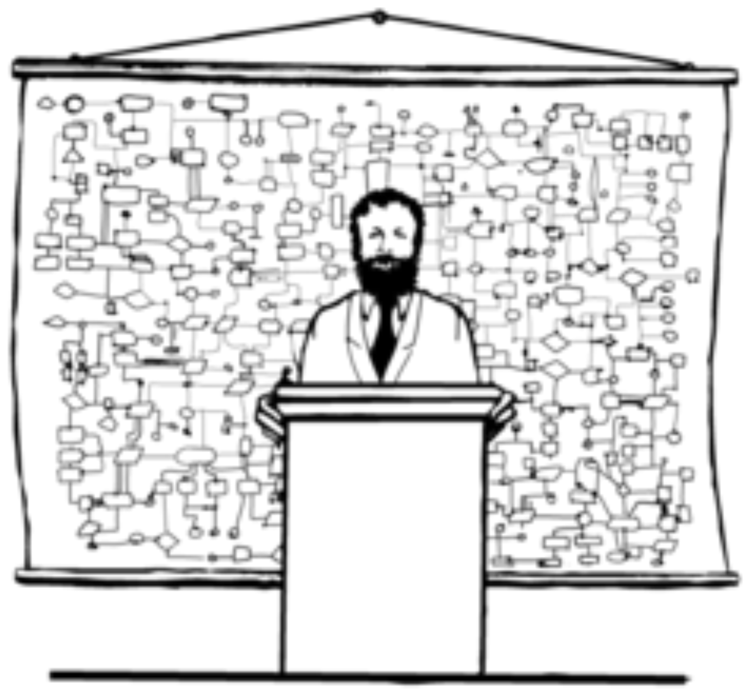
Introduction to Computer Science

CSCI 109

Readings

St. Amant, Ch. 3



"Now that you have an overview of the system, we're ready for a little more detail"

Andrew Goodney

Fall 2017

Lecture 2: Architecture *Aug. 28, 2017*

Reminders

- Take the survey if you haven't already (instructions on bytes.usc.edu/cs109/)
- Work on HW #1. It is due 9/11.
- Quiz #1 is on 9/25. It will cover material taught on 8/21 and 8/28.

Schedule

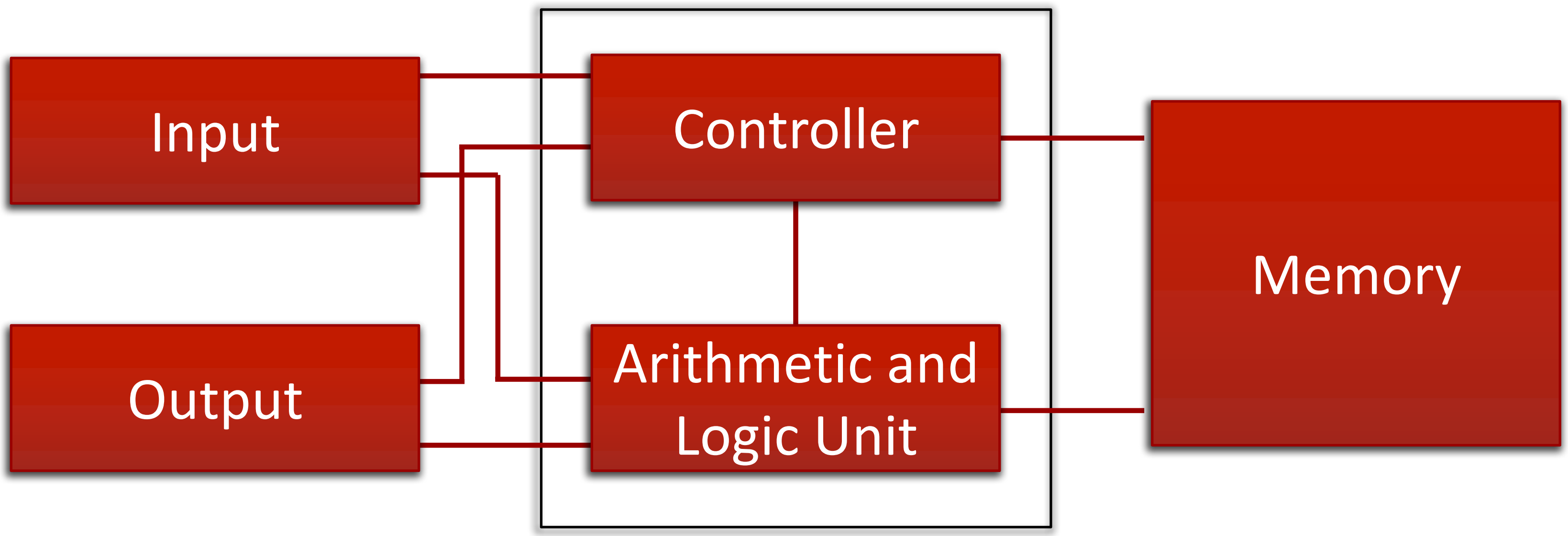
Date	Topic		Assigned	Due	Quizzes/Midterm/Final
21-Aug	Introduction	What is computing, how did computers come to be?			
28-Aug	Computer architecture	How is a modern computer built? Basic architecture and assembly	HW1		
4-Sep	Labor day				
11-Sep	Data structures	Why organize data? Basic structures for organizing data		HW1	
18-Sep	Last day to drop a Monday-only class without a mark of “W” and receive a refund or change to Pass/No Pass or Audit for Session 001				
25-Sep	Data structures	More structures for organizing data	HW2		**Quiz 1** on material taught in class 8/21-8/28
2-Oct	Algorithms	Operating on data, the notion of efficiency, simple algorithms			**Quiz 2** on material taught in class 9/11-9/25
6-Oct	Last day to drop a course without a mark of “W” on the transcript				
9-Oct	Algorithms and programming	(Somewhat) More complicated algorithms and simple programming constructs		HW2	**Quiz 3** on material taught in class 10/2
16-Oct	Operating systems	What is an OS? Why do you need one?	HW3		**Quiz 4** on material taught in class 10/9
23-Oct	Midterm	Midterm			**Midterm** on all material taught so far.
30-Oct	Computer networks	How are networks organized? How is the Internet organized?		HW3	
6-Nov	Artificial intelligence	What is AI? Search, plannning and a quick introduction to machine learning	HW4		**Quiz 5** on material taught in class 10/30
10-Nov	Last day to drop a class with a mark of “W” for Session 001				
13-Nov	The limits of computation	What can (and can't) be computed?			**Quiz 6** on material taught in class 11/6
20-Nov	Robotics	Robotics: background and modern systems (e.g., self-driving cars)		HW4	**Quiz 7** on material taught in class 11/13
27-Nov	Summary, recap, review	Summary, recap, review for final			**Quiz 8** on material taught in class 11/20
8-Dec	Final exam 11 am - 1 pm in SAL 101				**Final** on all material covered in the semester

Computer Architecture

- The von Neumann architecture
- The Central Processing Unit (CPU)
- Storage
- Input and Output

Reading:
St. Amant   Ch. 3

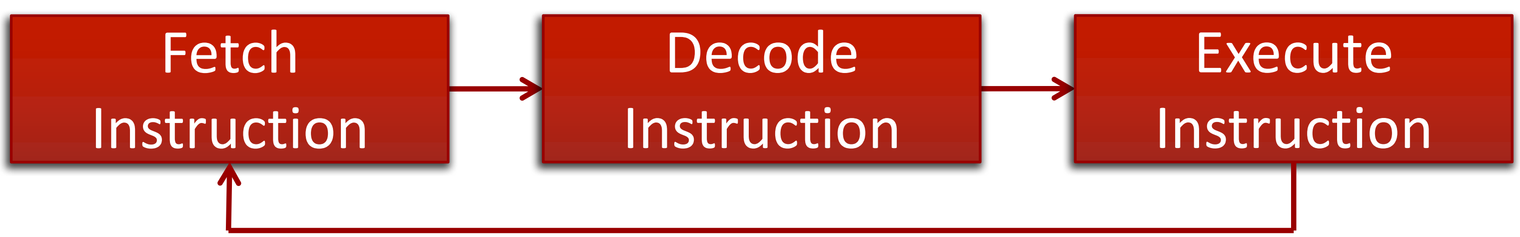
The von Neumann Architecture



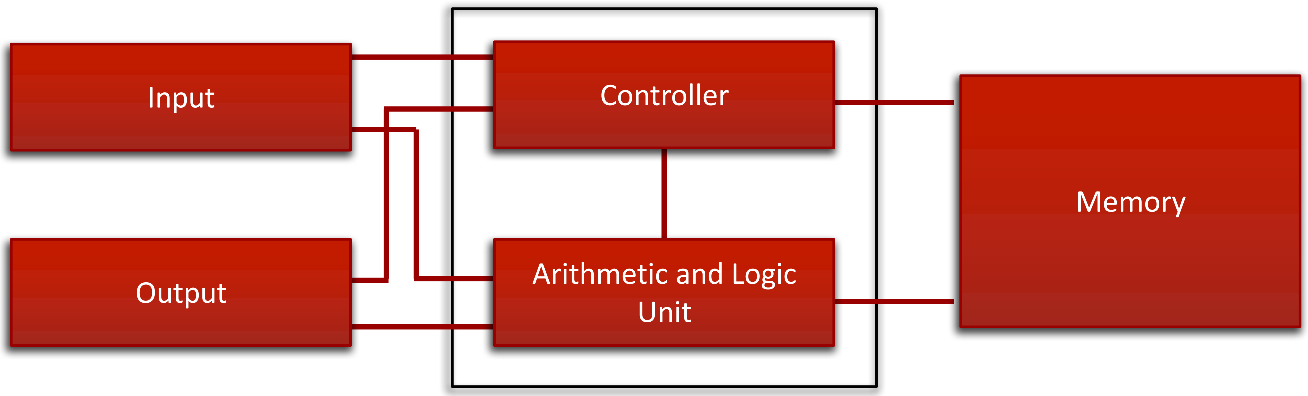
Controller + ALU = Central Processing Unit (CPU)

The Central Processing Unit (CPU)

- Controller + ALU = Central Processing Unit (CPU)
- CPU has a small amount of temporary memory within it
  - Registers
  - A special register called the program counter (PC)
- CPU performs the following cycle repeatedly

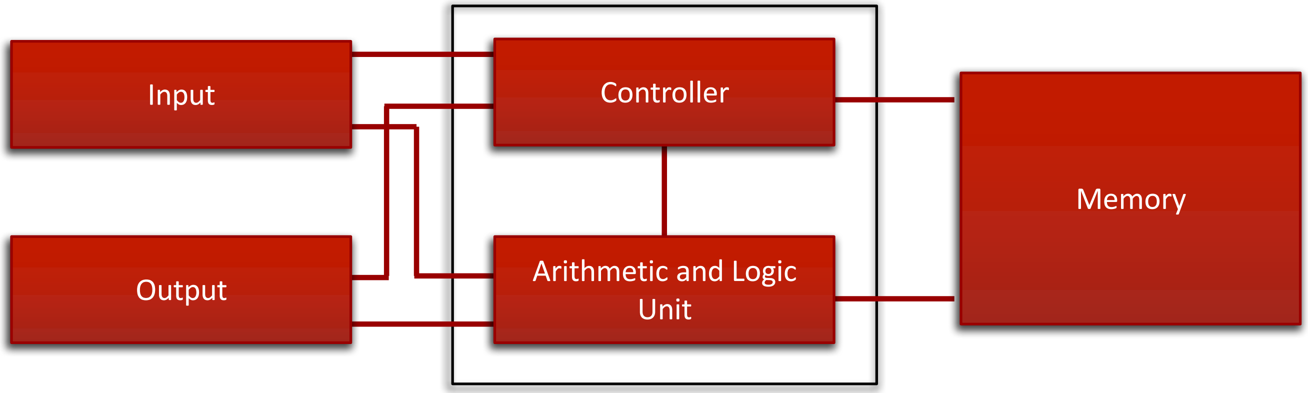


Typical Controller Tasks



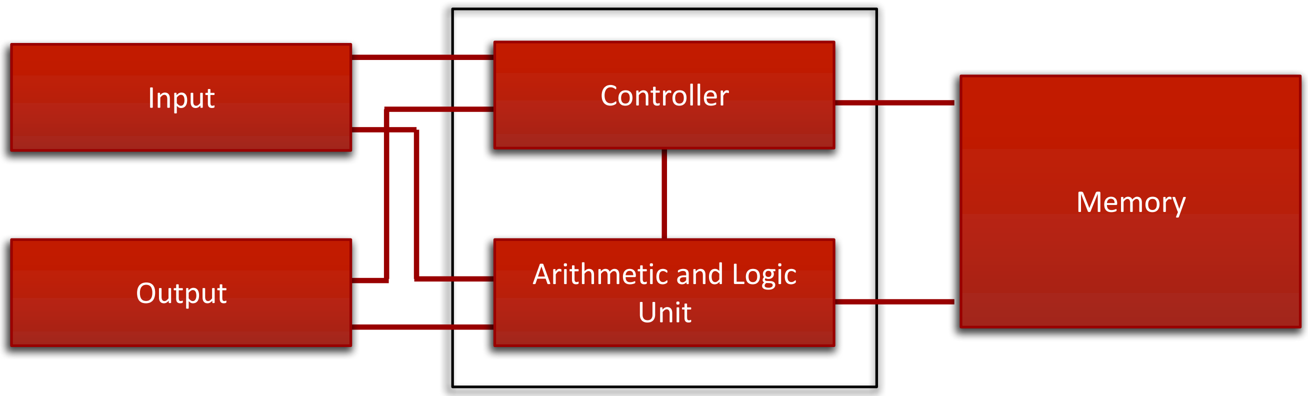
- Read an instruction from memory
- Direct ALU to do some arithmetic or logic operation
- Transfer data from one place to another
- Prepare for next instruction to be read
- Send a directive to input or output device

Typical ALU Tasks



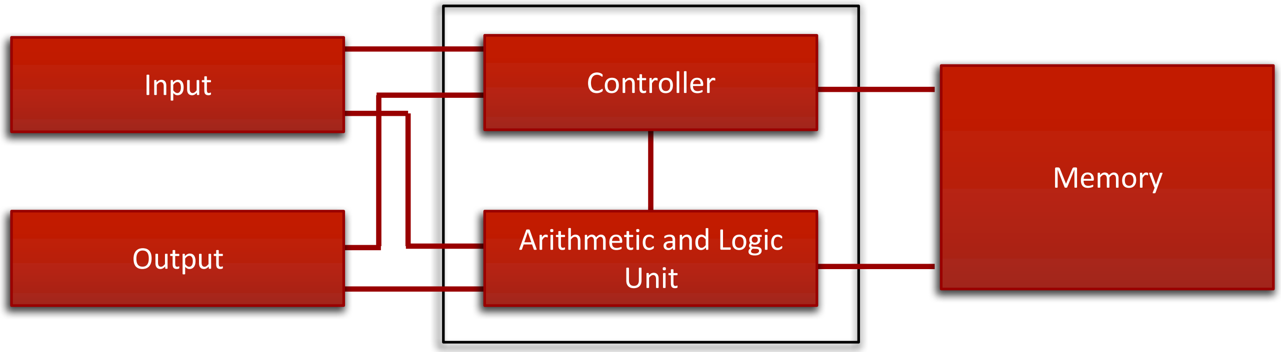
- Perform an arithmetic operation on the contents of registers e.g., add R1 and R2 and put the result in R1 (R1 = R1 + R2)
- Perform a logical operation on the contents of registers e.g., compare R1 and R2 (R1 < R2 ?)

Improving the CPU



- Faster clock (~GHz on modern computers)
- Specialized ALU (e.g., GPU) or more than one ALU
- Multiprocessing (more than one CPU)
- Streamlining Controller-ALU cooperation (pipelining)
- More complex ALU instructions ?

Storage: Addressing and Random Access



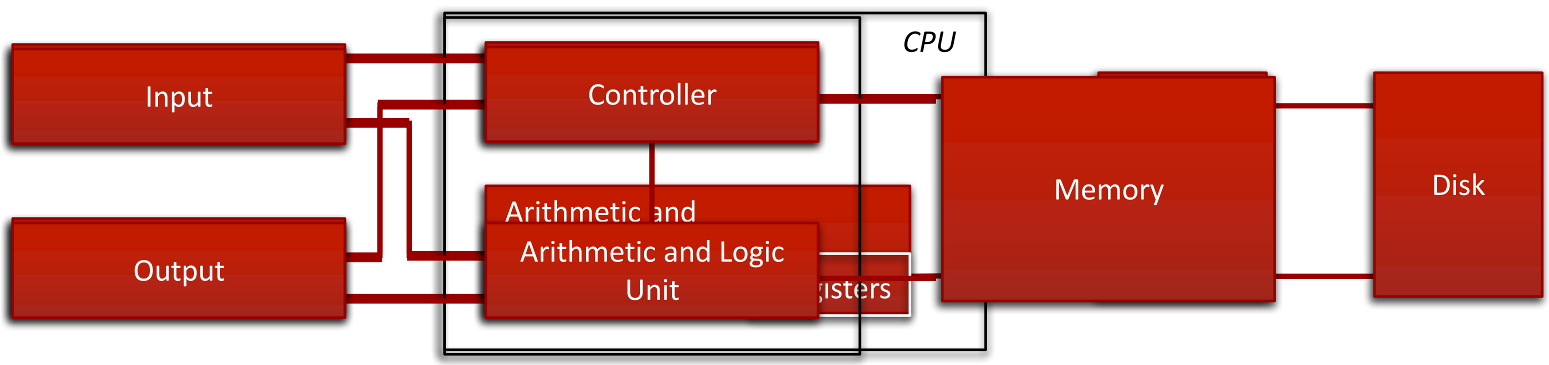
- How is storage organized ?
- Linear ordering
  - Each stored item has a number (an address)
  - To retrieve an item you have to know its number
  - Retrieval is by <u>address</u>

Storage: Modular and Hierarchical

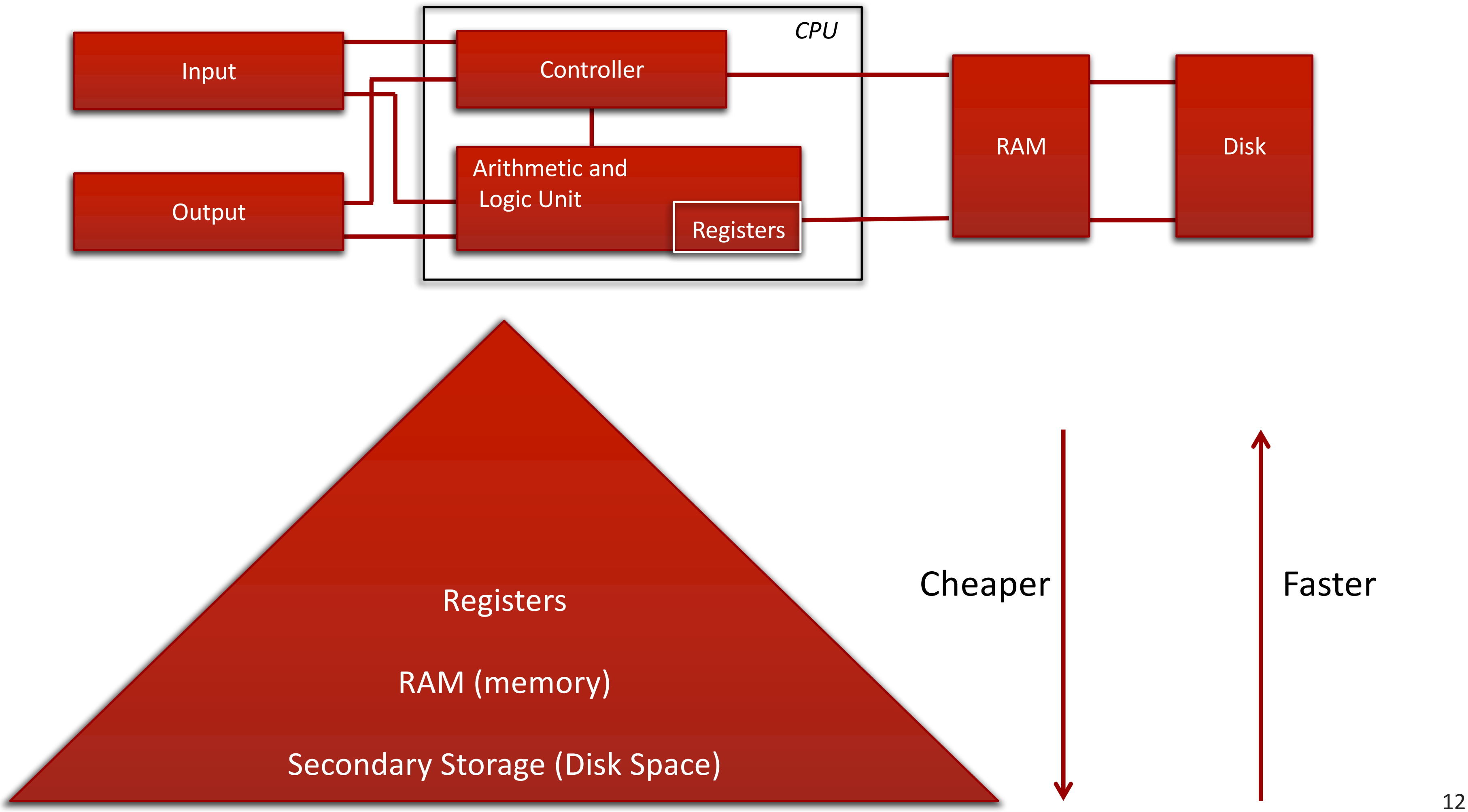
- Analogy with office files
  - Each (physical) file has a number
  - When a file is needed, you ask for it by number
  - In the office, files may be stored in shelves, on tables, in cabinets, etc.
- Storage is *modular*: as long as the person who wants the file knows its number and as long as the storekeeper can find the file, doesn't matter exactly how physical files are stored
- Or where they are stored: Files may be stored in a basement – not easy to reach

The Nature of Computer Storage

- Only one item can be stored at one memory location
- Random access (RAM) – all locations in memory are (on the average) equally slow (or fast) in terms of access speed
- Storage is hierarchical

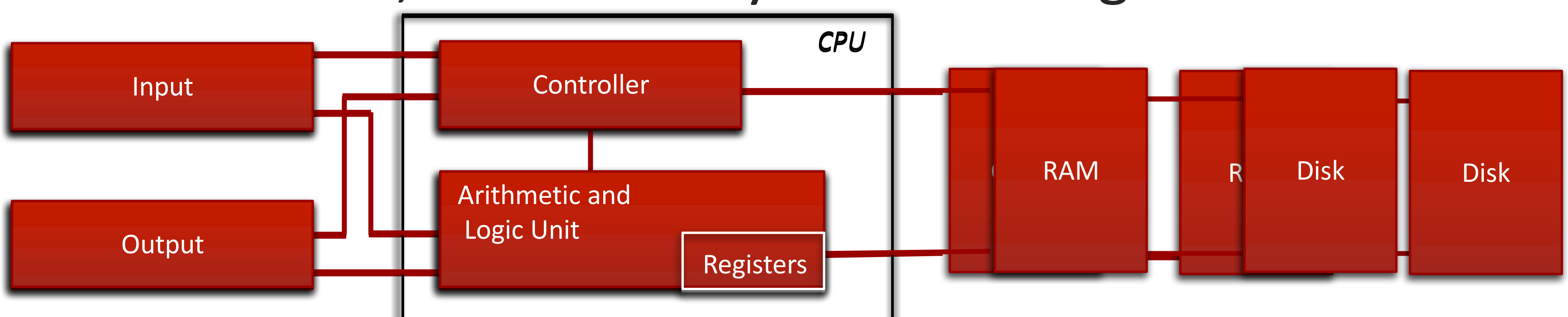


The Storage Hierarchy



Volatile Storage, Special Memory

- Volatile: memory is erased when power turned off
  - Registers, RAM (often called primary storage)
- Non-volatile: memory intact when power turned off
  - Secondary storage (disk)
- Booting and ROM (read-only memory)
- Cache: small, fast memory between registers and RAM

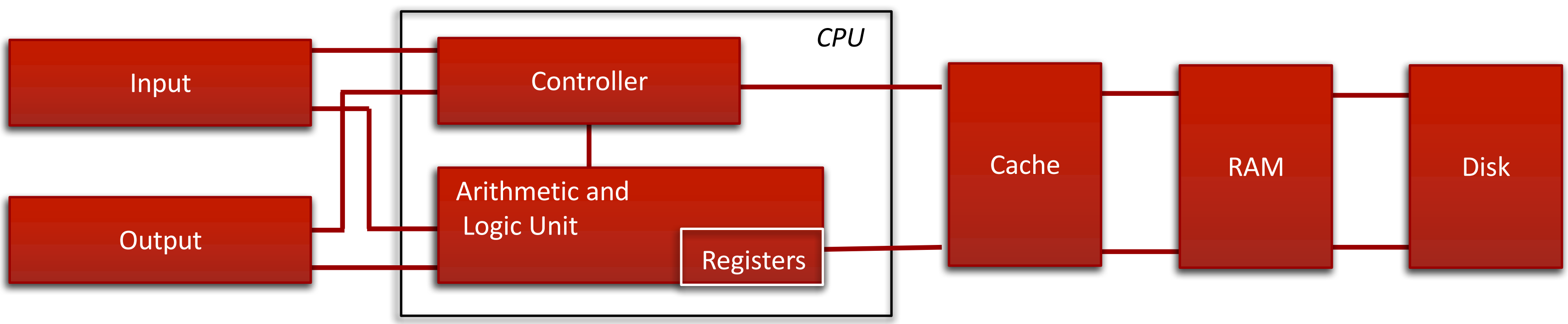


Cache

- Small (but bigger than registers)
- Volatile
- Fast (not as fast as registers, but faster than RAM)
- What to keep in the cache ?
  - Things that programs are likely to need in the future
  - Locality principle:
    - Look at what items in memory are being used
    - Keep items from nearby locations (spatial locality)
    - Keep items that were recently used (temporal locality)

Input and Output

- Input: Interrupt-driven
  - e.g., Key strokes are slow, CPU treats them like special events
- Output: Write to special memory (e.g., video memory)



Creating Assembly (Machine) Code

- Adding four numbers in C is easy

a = b + c + d + e

- Equivalent assembly code

add a, b, c	OR	add a, b, c
add a, a, d		add f, d, e
add a, a, e		add a, a, f

- Which one is better ? Are the two equivalent ?

Programs at Different Abstraction Levels

- C code a = b + c
- Assembly code add a, b, c
- Machine code 00000010001100100100000000100000

Programs at Different Abstraction Levels

- C code a = b + c
- Assembly code add a, b, c
- Machine code 000000|10001|10010|01000|00000|100000

6 bits – opcode

5 bits: source register 1

5 bits: source register 2

5 bits: destination register

5 bits: shift amount

5 bits: functions

Typical Operations

◆ ADD Ri Rj Rk	Add contents of registers Ri and Rj and put result in register Rk
◆ SUBTRACT Ri Rj Rk	Subtract register Rj from register Ri and put result in register Rk
◆ AND Ri Rj Rk	Bitwise AND contents of registers Ri, Rj and put result in register Rk
◆ NOT Ri	Bitwise NOT the contents of register Ri
◆ OR Ri Rj Rk	Bitwise OR the contents of registers Ri, Rj and put result in register Rk
◆ SET Ri value	Set register Ri to given value
◆ SHIFT-LEFT Ri	Shift bits of register Ri left
◆ SHIFT-RIGHT Ri	Shift bits of register Ri right
◆ MOVE Ri Rj	Copy contents from register Ri to register Rj
◆ LOAD Mi Ri	Copy contents of memory location Mi to register Ri
◆ WRITE Ri Mi	Copy contents of register Ri to memory location Mi
◆ GOTO Mi	Jump to instruction stored in memory location Mi
◆ COND_GOTO Ri Rj Mi	If Ri > Rj and R2, jump to instruction stored in memory location Mi

A Typical Day at PaniCorp

- Central processing: Connie and Alun
- The bus: Buster
- Storage (Memory): Mr. Lager
- Input and output

Read St. Amant Ch. 3

Play out a typical instruction cycle with your friends

```
M100    SET R1 MI
M101    SET R2 0
M102    SET R3 1
M103    SET R6 0
M104    ADD R1 R2 R4
M105    SUB  R1 R3 R5
M106    MOVE R5 R1
M107    MOVE R4 R2
M108    COND_GOTO R1 R6 104
M109    WRITE R2 M2
M110    END
```

What does this program do?

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	

```
M100    SET R1 MI
M101    SET R2 0
M102    SET R3 1
M103    SET R6 0
M104    ADD R1 R2 R4
M105    SUB  R1 R3 R5
M106    MOVE R5 R1
M107    MOVE R4 R2
M108    COND_GOTO R1 R6 104
M109    WRITE R2 M2
M110    END
```

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	

M100	SET R1 MI
M101	SET R2 0
M102	SET R3 1
M103	SET R6 0
M104	ADD R1 R2 R4
M105	SUB R1 R3 R5
M106	MOVE R5 R1
M107	MOVE R4 R2
M108	COND_GOTO R1 R6 104
M109	WRITE R2 M2
M110	END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	
M105	1	5	1	6	0	0	3	

M100	SET R1 MI
M101	SET R2 0
M102	SET R3 1
M103	SET R6 0
M104	ADD R1 R2 R4
M105	SUB R1 R3 R5
M106	MOVE R5 R1
M107	MOVE R4 R2
M108	COND_GOTO R1 R6 104
M109	WRITE R2 M2
M110	END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	
M105	1	5	1	6	0	0	3	
M106	0	5	1	6	0	0	3	

M100	SET R1 MI
M101	SET R2 0
M102	SET R3 1
M103	SET R6 0
M104	ADD R1 R2 R4
M105	SUB R1 R3 R5
M106	MOVE R5 R1
M107	MOVE R4 R2
M108	COND_GOTO R1 R6 104
M109	WRITE R2 M2
M110	END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	
M105	1	5	1	6	0	0	3	
M106	0	5	1	6	0	0	3	
M107	0	6	1	6	0	0	3	

M100	SET R1 MI
M101	SET R2 0
M102	SET R3 1
M103	SET R6 0
M104	ADD R1 R2 R4
M105	SUB R1 R3 R5
M106	MOVE R5 R1
M107	MOVE R4 R2
M108	COND_GOTO R1 R6 104
M109	WRITE R2 M2
M110	END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	
M105	1	5	1	6	0	0	3	
M106	0	5	1	6	0	0	3	
M107	0	6	1	6	0	0	3	
M108	0	6	1	6	0	0	3	

M100	SET R1 MI
M101	SET R2 0
M102	SET R3 1
M103	SET R6 0
M104	ADD R1 R2 R4
M105	SUB R1 R3 R5
M106	MOVE R5 R1
M107	MOVE R4 R2
M108	COND_GOTO R1 R6 104
M109	WRITE R2 M2
M110	END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	
M105	1	5	1	6	0	0	3	
M106	0	5	1	6	0	0	3	
M107	0	6	1	6	0	0	3	
M108	0	6	1	6	0	0	3	
M109	0	6	1	6	0	0	3	6

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	
M105	1	5	1	6	0	0	3	
M106	0	5	1	6	0	0	3	
M107	0	6	1	6	0	0	3	
M108	0	6	1	6	0	0	3	
M109	0	6	1	6	0	0	3	6
M110	0	6	1	6	0	0	3	6

M100 SET R1 MI
M101 SET R2 0
M102 SET R3 1
M103 SET R6 0
M104 ADD R1 R2 R4
M105 SUB R1 R3 R5
M106 MOVE R5 R1
M107 MOVE R4 R2
M108 COND_GOTO R1 R6 104
M109 WRITE R2 M2
M110 END

PC	R1	R2	R3	R4	R5	R6	M1	M2
							3	
M100	3						3	
M101	3	0					3	
M102	3	0	1				3	
M103	3	0	1			0	3	
M104	3	0	1	3		0	3	
M105	3	0	1	3	2	0	3	
M106	2	0	1	3	2	0	3	
M107	2	3	1	3	2	0	3	
M108	2	3	1	3	2	0	3	
M104	2	3	1	5	2	0	3	
M105	2	3	1	5	1	0	3	
M106	1	3	1	5	1	0	3	
M107	1	5	1	5	1	0	3	
M108	1	5	1	5	1	0	3	
M104	1	5	1	6	1	0	3	
M105	1	5	1	6	0	0	3	
M106	0	5	1	6	0	0	3	
M107	0	6	1	6	0	0	3	
M108	0	6	1	6	0	0	3	
M109	0	6	1	6	0	0	3	6
M110	0	6	1	6	0	0	3	6

What does this program do?

Input is a number (say n) stored in M1

Output is a number (say S) stored in M2

where $S = 1 + 2 + 3 + 4 \ldots + n$

Try it out beginning with other values stored in M1