CSD101: Introduction to computing and programming (ICP)

A very simple computer(VSC) I

- An **accumulator** is a register that holds one data element (i.e. an integer, real number or character (one or more depending on the encoding)). Their size is 2/ 4/ 8 bytes. Modern computers have 64-bit or 8-byte registers.
- **PC** (program counter) contains the address of the instruction to be executed next.
- Random access memory (RAM) addresses start from 1 and increase by 1 (byte). Access is typically in chunks of 4/ 8 bytes at a time.
- In principle the very simple computer is as powerful as any modern day computer.
- A VSC can be hard to program because it has a very simple, small set of operations on data (later slide).

A very simple computer(VSC) II

- For our purposes we assume that the VSC works in terms of units (decimal numbers and characters) and representations (internal representations of the current OS) that are convenient. For simplicity VSC works only with whole numbers.
- Memory size of VSC is 1000 units - addresses from 0 to 999.

Instruction set for the very simple computer I

Note: A is an address; acc stands for accumulator; @A means contents of address A; PC is program counter, N - number, S - string

Instrn code	**Instrn format**	**Meaning**
0	halt	Program halts
1	strt	Start of program
2	lodm A	acc = @A (load from memory)
3	lodn N	acc = N (acc loaded with number N)
4	lods S	acc = S (acc loaded with string S)
5	stor A	store acc at addr A
6	add A	acc = acc + @A
7	neg A	acc = -acc
8	jmp A	PC = A
9	jmp- A	if (acc<0) PC = A
10	jmp0 A	if (acc==0) PC = A
11	jmp+ A	if (acc>0) PC = A
12	inpn	acc = number read from input device
13	inps	acc = string read from input device
14	out	write acc to output device (string)
15	outl	write new line to output device

Writing code for VSC

Code for VSC is in the form of a sequence of lines of code. Each line is stored in successive addresses in memory or RAM. Each line has the following format:

Address	Operator code	Optional operand

The fields should be separated by one or more spaces. The lines of the program should always be stored in successive addresses else it is an error. To comment VSC code we use the # symbol. Anything after the # till the end of line is neglected.

Some simple problems

1. Output a message.
2. Read two numbers m and n and output the larger number.
3. Read two numbers m and n and output the product of the two numbers.
4. Read two positive integers m and n and output m^n.
5. Read two positive integers m and n and output the largest number that divides both m and n.

We write VSC code for the first three.
Problems 4 to 5 are for you to practice in lab 1, Q3.

Example of VSC program - 1

Example 1

A VSC program that prints `Very Simple Computer`.

Address	**Instn. code**	**Optional operand**	**Comment**
0	*1*		*#start*
1	*4*	*"Very Simple Computer"*	*#load string in acc*
2	*14*		*#output contents of acc*
3	*15*		*#output newline*
4	*0*		*#halt*

Example of VSC program - 2

Example 2

A VSC program that reads two numbers and prints the larger of the two numbers.

Address	Instn. code	Optional operand	Comment
0	1		#start
1	12		#Read first number, acc=n1
2	5	500	#Store n1 at addr 500
3	12		#Read second number, acc=n2
4	5	501	#Store n2 at 501
5	7		#n2 is negated, acc=-n2
6	6	500	#add n1, acc=n1-n2
7	11	14	#n1 is larger, jmp+ to 14
8	2	501	#n2 $\geq$ n1, load n2, acc=n2
9	14		#output acc
10	4	" is larger or equal."	#load string, acc="is larger or equal"
11	14		#output string
12	15		#output newline
13	0		#halt
14	2	500	#n1 $>$ n2, load n1, acc=n1
15	8	9	#jump to addr 9

Example of VSC program - 3

Read m, n output $m \times n$. Has to be done by repeated addition.

Pseudo code:

Line no.	**Pseudocode**	**Comment**
0		# m, n are non-negative numbers.
1	read m	
2	read n	
3	p=0	#p stores the product
4	if (m equals 0) go to line 8	#product calculation done.
5	p=p+n	
6	m=m-1	
7	go to line 4	
8	output p	
9	output newline	
10	stop	

VSC code example - 3

Address	Instn code	Opt. operand	Comment
0	1		#start
1	12		#read m, acc=m
2	5	500	#store m at addr 500
3	12		#read n, acc=n
4	5	501	#store n at addr 501
5	3	0	#load acc with 0
6	5	502	#store at 502, product p=0
7	3	-1	#load constant -1
8	5	503	#-1 stored at addr 503
9	2	500	#load m, acc=m
10	10	18	#jump to 18 if acc is 0. Mult done.
11	2	502	#load p, acc=p
12	6	501	#add n to p
13	5	502	#store p at 502
14	2	500	#load m in acc
15	6	503	#m=m-1
16	5	500	#store m at addr 500
17	8	10	#jump to addr 10
18	2	502	#load acc with p - the product
19	14		#output the product
20	15		#output newline
21	0		#stop

Example of VSC program - 4

- Read m, n and output m^n.
- Has to be done by repeated multiplication.
- We have to reuse the multiplication program of example 3 repeatedly.
- The pseudocode is written in the next slide. (Convert it to VSC code - lab 1 Q3 exercise)

Pseudocode for example 4

Line no.	Pseudocode	Comment
1		# m, n are positive integers.
2	read n	
3	read m	
4	r=1	#r is the result
5	if (n equals 0) go to line 9	
6	r=m*r	#reuse multiplication using fresh addresses
7	n=n-1	
8	go to line 5	
9	output r	
10	stop	

Algorithms and programs

- A systematic way to solve a problem is called an **algorithm**.

Definition 1

An algorithm is a systematic, finite, effective, step-by-step method to solve a problem.

  - `Systematic: correct answer for all inputs`
  - `Finite: terminates in finite time for all inputs`
  - `Effective: we know exactly how to do every step`

- Pseudocode or programs in any programming language implement algorithms (most of the time).

Example of VSC program - 5

- To get the greatest common divisor (GCD) of two positive integers m, n we must first figure out how to solve the problem. That is we need an algorithm.
- The algorithm to get the GCD is based on the following property of integers. If $z > 0$ is the greatest common divisor of m and n with $m > n$ then z will also divide $m - n$. Now we can repeat this process of $larger - smaller$ with $m - n$ and n until $m = n$ at which point we have the GCD - it is m. It is clear this will terminate since we calculate $larger - smaller$ in each step and the difference keeps decreasing.

Example 3

m	**n**	$m - n$ (*assuming* $m > n$)
64	*36*	*28*
36	*28*	*8*
28	*8*	*20*
20	*8*	*12*
12	*8*	*4*
8	*4*	*4*
4	*4*	$m = n$, *terminates, GCD=4*