UNIVERSITY OF PRIMORSKA
Faculty of Mathematics, Natural Sciences and Information Technologies
Department of Information Sciences and Technology

Systems 1

Tutorials in assembly language

Author: Domen Šoberl

2022-2023

Contents

1 Numbers ...

1.1 Converting between number systems

Convert the following decimal numbers first to 16-bit binary and then to hexadecimal representation:

a) 67 b) 49155 c) 65534

Convert the following 16-bit hexadecimal numbers first to binary and then to decimal representation:

a) 1000 b) ABCD c) FFFF

1.2 Number spaces

How many different numbers can we write with this many bits?

a) 8 b) 10 c) 16 d) 20 e) 32

How many bits do we need, to address:

a) 14 B b) 64 KB c) 1 MB d) 100 MB e) 1 GB

Suppose we have a 16-bit addressing space. Divide this space into 4 blocks of equal sizes. Write down the first and the last address of each block. *We will write hexadecimal addresses with the prefix* `0x`.

How many addresses are between `0x1000` and `0x2000` including the starting and the ending address?

We want to store 1000 numbers, each number at its own address, starting with the address `0x0200`. What is the address of the last number?

1.3 Computer arithmetic

Calculate the sums of hexadecimal numbers:

a) A + B b) A781 + 1942 c) A000 + 7000

Suppose we operate with 16-bit numbers only. What is then the result of addition A000 + 7000?

In the 16-bit number space let $x =$ `0xA123`. Which number in that space is equivalent to $-x$, i.e. for which number $y = -x$ it holds $x + y = 0$?

2 Introduction to assembly language

Task 2.1 Explain the meaning and the function of all the registers of our simulated computer system.

Solution:

A, B, C, D	General purpose registers for performing arithmetic logic operations.
IP	*Instruction pointer* - memory location of the next instruction.
SP	*Stack pointer* - memory location of the top of the stack
SR	*Status register* - all status flags

Status flags:

M	*Interrupt mask* - globally enables/disables interrupts.
C	*Carry* - set when an overflow occurs after an arithmetic operation.
Z	*Zero* - set when the result of the arithmetic operation is 0.
F	*Fault* - set when there an error occurs during program execution.
H	*Halt* - set after the program execution has stopped.

Task 2.2 Compile and run the following assembly program:

```
1 MOV A, 0x10
2 ADD A, 10
3 HLT
```

Write down and explain the compiled machine code. Trace the program step by step and explain its behavior.

Solution:

Compiler produced the following machine code: `06 00 00 10 14 00 00 0A 00`.
Explanation:

```
MOV [06] A [00], 0x10 [00 10]
ADD [14] A [00], 10 [00 0A]
HLT [00]
```

The program moves hexadecimal number 10 to the 16-bit register A and adds to it decimal number 10.

Task 2.3 Move $AL \leftarrow 120$ and $BL \leftarrow 180$. Do the 8-bit and the 16-bit addition of those values. Explain the different outcomes.

Solution:

```
; 8-bit addition          ; 16-bit addition
MOVB AL, 120              MOVB AL, 120
MOVB BL, 180              MOVB BL, 180
ADDB AL, BL               ADD A, B
HLT                       HLT
```

8-bit addition results in $A = 44$ and 16-bit in $A = 300$. With 8-bit addition the overflow occurs and the carry bit (C) is set.

Task 2.4 Two 8-bit values x and y are stored in the 16-bit register A, so that $AH = x$ and $AL = y$. Write a program that moves these values so that $A = x$ and $B = y$. Use AND masking.

Solution:

```
MOV B, A
SHR A, 8
AND B, 0x00FF
```

Prepare for the exam:

a) 16-bit registers A and B hold 8-bit values x and y respectively. Write a program that moves these values so that $AH = x$ and $AL = y$.

b) Do the same thing by using only 16-bit instructions and OR operation.

c) What does the following program in machine code do: `0x10 0x10 0x00 0x41`?

d) Run the below assembly program and observe the status register after the execution has finished. Explain why each status bit has changed.

```
MOV A, 0xFF00
ADD A, 0x0100
HLT
```

3 Memory addressing and variables

Review the different addressing modes listed in Appendix A.3. Remember:

- Square brackets `[ ]` represent memory access.
- Every CPU instruction can make at most one access to memory.

Task 3.1 What is the difference between programs:

```
MOV D, 100
MOV A, D
```

and

```
MOV D, 100
MOV A, [D]
```

Task 3.2 Write the hexadecimal value `0xABCD` to memory address `0x0100`. Which addresses store which parts of this value? Now write the 8-bit number `0x33` to memory address `0x0102`. Write a program that adds the value on address `0x0102` to the value on address `0x0100`.

Task 3.3 Define a 16-bit variable x with initial value $0xABCD$. Then increase the value of this variable by 3. Define the variable x in the following ways:

a) as a fixed memory address `0x0100`,
b) as a label below the program code,
c) as a label above the program code,
d) as a label at memory address `0x0100`.

Task 3.4 Define 16-bit variables x, y, and z. Write a program that computes $z = 3z - (x + y)/2$.

Prepare for the exam:

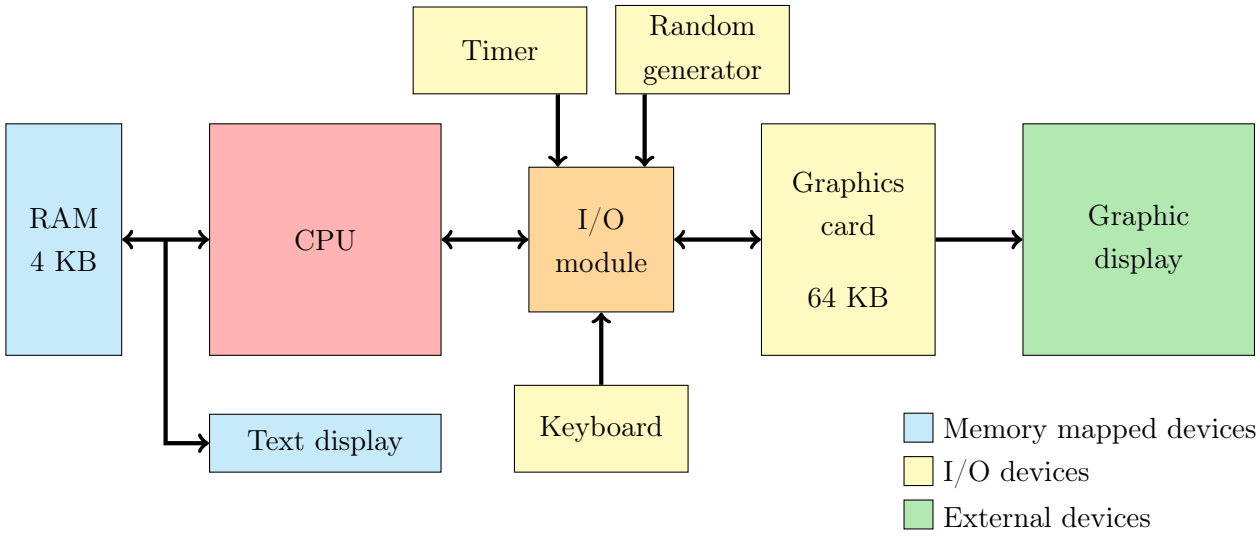
a) Define variables x and y with the initial values of your choice. Write a program that switches the values of x and y.

b) Write a program that computes $z = 16 \cdot (2x - y)$. Use bit shifting for multiplication.

c) Write a program that computes $z = x^2 - y^2$. Use instruction `MUL` to square a value.

A Architecture of the simulated system

A.1 System architecture



A.2 CPU registers

16-bit registers:

Register	Description	Index
A	General purpose register	0
B	General purpose register	1
C	General purpose register	2
D	General purpose register	3
SP	Stack Pointer	4
IP	Instruction Pointer	5
SR	Status Register	6

8-bit registers:

Register	Description	Index
AH	Higher part of register A	7
AL	Lower part of register A	8
BH	Higher part of register B	9
BL	Lower part of register B	10
CH	Higher part of register C	11
CL	Lower part of register C	12
DH	Higher part of register D	13
DL	Lower part of register D	14

A.3 Addressing modes

Addressing mode	Abbreviation	Example
Immediate	IMD	`ADD A, 100`
Register	REG	`ADD A, B`
Direct	DIR	`ADD A, [100]`
Indirect	IND	`ADD A, [B]`

A.4 Instruction set

CPU instruction format:

opcode	operand 1 (optional)	operand 2 (optional)

CPU instruction set:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	HLT	MOV REG,REG	MOV REG,IND	MOV REG,DIR	MOV IND,REG	MOV DIR,REG	MOV REG,IMD	MOV IND,IMD	MOV DIR,IMD	MOVB REG,REG	MOVB REG,IND	MOVB REG,DIR	MOVB IND,REG	MOVB DIR,REG	MOVB REG,IMD	MOVB IND,IMD
1	MOVB DIR,IMD	ADD REG,REG	ADD REG,IND	ADD REG,DIR	ADD REG,IMD	ADDB REG,REG	ADDB REG,IND	ADDB REG,DIR	ADDB REG,IMD	SUB REG,REG	SUB REG,IND	SUB REG,DIR	SUB REG,IMD	SUBB REG,REG	SUBB REG,IND	SUBB REG,DIR
2	SUBB REG,IMD	INC REG	INCB REG	DEC REG	DECB REG	CMP REG,REG	CMP REG,IND	CMP REG,DIR	CMP REG,IMD	CMPB REG,REG	CMPB REG,IND	CMPB REG,DIR	CMPB REG,IMD	JMP IND	JMP DIR	JC IND
3	JC DIR	JNC IND	JNC DIR	JZ IND	JZ DIR	JNZ IND	JNZ DIR	JA IND	JA DIR	JNA IND	JNA DIR	PUSH REG	PUSH IMD			PUSHB REG
4	PUSHB IMD			POP REG	POPB REG	CALL IND	CALL DIR	RET	MUL REG	MUL IND	MUL DIR	MUL IMD	MULB REG	MULB IND	MULB DIR	MULB IMD
5	DIV REG	DIV IND	DIV DIR	DIV IMD	DIVB REG	DIVB IND	DIVB DIR	DIVB IMD	AND REG,REG	AND REG,IND	AND REG,DIR	AND REG,IMD	ANDB REG,REG	ANDB REG,IND	ANDB REG,DIR	ANDB REG,IMD
6	OR REG,REG	OR REG,IND	OR REG,DIR	OR REG,IMD	ORB REG,REG	ORB REG,IND	ORB REG,DIR	ORB REG,IMD	XOR REG,REG	XOR REG,IND	XOR REG,DIR	XOR REG,IMD	XORB REG,REG	XORB REG,IND	XORB REG,DIR	XORB REG,IMD
7	NOT REG	NOTB REG	SHL REG,REG	SHL REG,IND	SHL REG,DIR	SHL REG,IMD	SHLB REG,REG	SHLB REG,IND	SHLB REG,DIR	SHLB REG,IMD	SHR REG,REG	SHR REG,IND	SHR REG,DIR	SHR REG,IMD	SHRB REG,REG	SHRB REG,IND
8	SHRB REG,DIR	SHRB REG,IMD	CLI	STI	IRET			IN REG	IN IND	IN DIR	IN IMD	OUT REG	OUT IND	OUT DIR	OUT IM	

A.5 Memory map

Address range	Component
`0x0000` – `0x0FFF`	RAM (4 KB)
`0x1000` – `0x101F`	Text display (2 lines × 16 ASCII characters)

A.6 Input / Output module

I/O Registers:

Register	Description	Index
IRQMASK	Enable/Disable specific interrupt requests	0
IRQSTATUS	Currently requested interrupts	1
IRQEIO	Clear a specific interrupt request	2
TMRPRELOAD	The initial timer value	3
TMRCOUNTER	The current timer value	4
KBDSTATUS	Keyboard status (a keypress has been detected)	5
KBDDATA	The data received from the keyboard	6
VIDMODE	Graphics card mode	7
VIDADDR	Address in the VRAM data	8
VIDDATA	Data at the VRAM address VIDADDR	9
RNDGEN	A randomly generated number	10

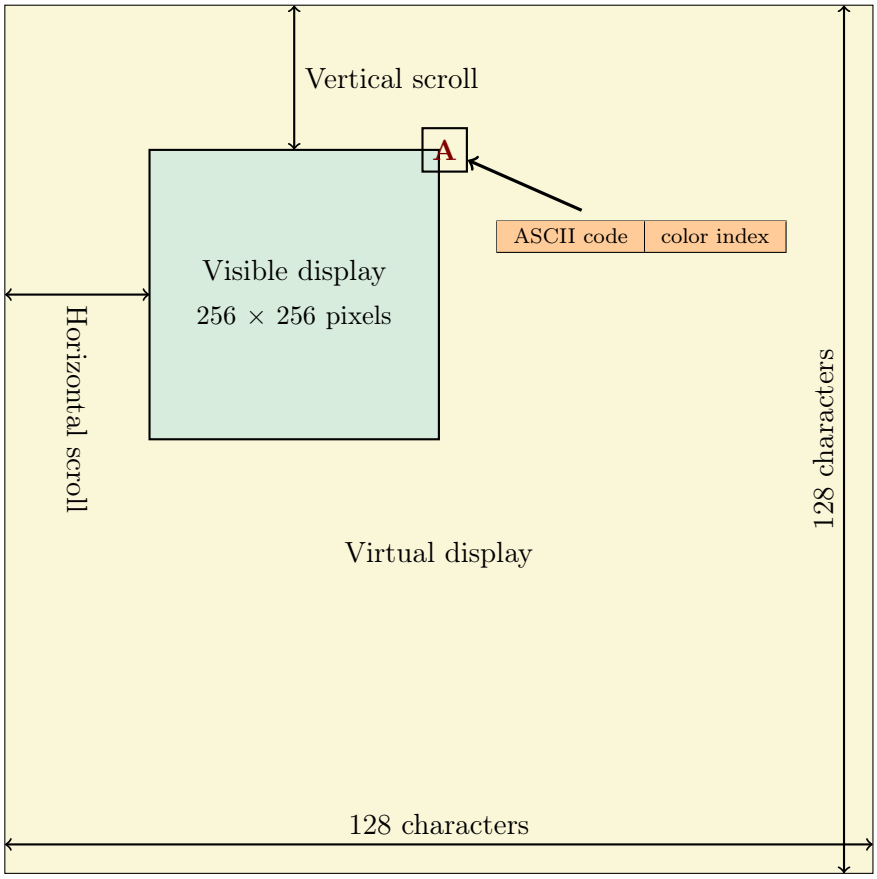
A.7 Graphics card

VIDMODE	Video mode	Description
0	DISABLED	Graphics card is disabled (default).
1	TEXT	Display text and custom-defined tiles.
2	BITMAP	Display individual pixels.
3	CLEAR	Clears the display. It works in both graphic modes.
4	RESET	Resets and disables the graphics card.

A.7.1 Text mode (VIDMODE = 1)

Text display properties:

- The size of each character is 16 × 16 pixels.
- Every displayed character has two properties: ASCII code (8-bit) and color index (8-bit).
- Default shape definitions of all 256 characters are stored in VRAM at addresses `0x8000` – `0x9FFF` and can be overwritten. By resetting the graphics card, the default values are restored.
- The standard RRRGGGBB palette is stored in VRAM at addresses `0xA000` – `0xA2FF` (3 bytes/color) and can be overwritten. By resetting the graphics card, the default values are restored.
- The size of the virtual display is 128 × 128 characters, but only a display window of 256 × 256 pixels is visible on the screen. The display window can be moved to achieve smooth scrolling effect.
- Eight characters can be placed on the visual display at any pixel position (256 × 256), independently of the virtual display content and scrolling state (sprite graphics).



VRAM Usage in text mode:

VRAM location	Size	Usage
`0x0000` – `0x7FFF`	32 KB	Virtual display area (128 × 128 characters, 2 bytes/character).
`0x8000` – `0x9FFF`	8 KB	Character set definition (256 characters, 32 bytes/character).
`0xA000` – `0xA2FF`	0.75 KB	Color palette (256 colors, 3 bytes/color).
`0xA300` – `0xA301`	2 B	Background color (color index 0 – 255, address `0xA300` is unused).
`0xA302` – `0xA303`	2 B	Horizontal scroll (display window x-offset in pixels).
`0xA304` – `0xA305`	2 B	Vertical scroll (display window y-offset in pixels).
`0xA306` – `0xA309`	4 B	Sprite 1 (character, color, x, y).
`0xA30A` – `0xA30D`	4 B	Sprite 2 (character, color, x, y).
`0xA30E` – `0xA311`	4 B	Sprite 3 (character, color, x, y).
`0xA312` – `0xA315`	4 B	Sprite 4 (character, color, x, y).
`0xA316` – `0xA319`	4 B	Sprite 5 (character, color, x, y).
`0xA31A` – `0xA31D`	4 B	Sprite 6 (character, color, x, y).
`0xA31E` – `0xA321`	4 B	Sprite 7 (character, color, x, y).
`0xA322` – `0xA325`	4 B	Sprite 8 (character, color, x, y).
`0xA326` – `0xFFFF`		Free memory (23754 bytes).

Character shape data (32 bytes):

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	`0x0000`
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	`0x0000`
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	`0x1FF8`
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	`0x3FFC`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0x3C3C`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0X3C3C`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0x3C3C`
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	`0x3FFC`
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	`0x3FFC`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0x3C3C`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0x3C3C`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0x3C3C`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0x3C3C`
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	`0x3C3C`
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	`0x0000`
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	`0x0000`

Virtual display area:

character 0	ASCII code	color index	32 KB
character 1	ASCII code	color index	
⋮	⋮		
character n	ASCII code	color index	

Character set definition:

ASCII code 0	shape data 0	8 KB
ASCII code 1	shape data 1	
⋮	⋮	
ASCII code 255	shape data 255	

Color palette definition:

color 0	red (8-bit)	green (8-bit)	blue (8-bit)	765 bytes
color 1	red (8-bit)	green (8-bit)	blue (8-bit)	
⋮		⋮		
color 255	red (8-bit)	green (8-bit)	blue (8-bit)	

Sprite data:

sprite 1	ASCII code	color	x	y	32 bytes
sprite 2	ASCII code	color	x	y	
⋮		⋮			
sprite 8	ASCII code	color	x	y	

A.7.2 Bitmap mode (VIDMODE = 2)

The entire 64 KB VRAM is used to display a bitmap image (256 × 256 pixels). The standard 8-bit color palette RRRGGGBB is used, which cannot be redefined.