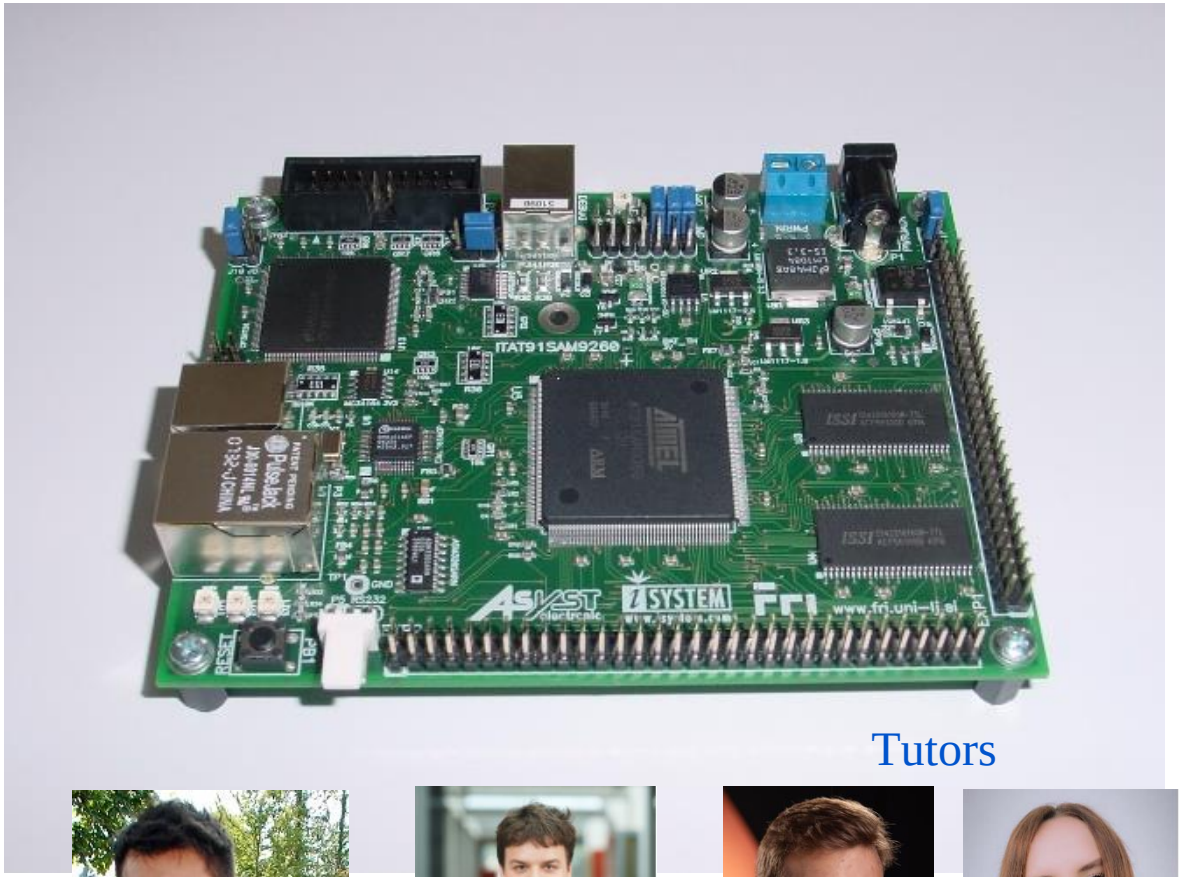
Computer architecture CA



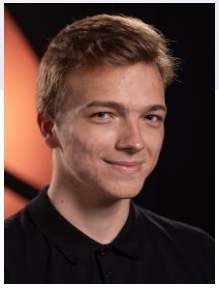
Team CA

Mira Trebar
mira.trebar@fri...

Žiga Pušnik
ziga.pusnik@fri....

Rok Češnovar
Rok.cesnovar@fri...

Tutors



Miha Krajnc
mk7793@student.uni-lj.si

Anamari Orehar
ao6477@student.uni-lj.si

Robert Rozman
rozman@fri.uni-lj.si

Computer architecture CA

LAB 1.1 General information

Laboratory exercises

- Learning the foundations of computer architecture from a practical view
- Understanding “How the computer works” by programming in ARM assembly language
- In-depth views:
  - computer operation
  - program execution
- Content upgrades: Computer Organization elective course and others

Content

- Basic knowledge needed from lectures (e.g. memory address, memory words, …)
- Core: Programming in ARM assembly language
- Format: lab exercises + 1 homework assignment
- 3 intermediate exams* (quizzes during lab sessions) - (november, december, january)
- Final exam preparations and exercises

- Alternative way: course seminar for advanced students – talk to instructor

**Due to Covid, can be changed*

Evaluation – grading*

- Lab marks represents 50% of the final mark for the course. You need to have:
  - successfully evaluated lab work (presence, work)
  - successfully evaluated homework assignment,
  - three intermediate evaluation exams (80 + 100 + 120 points)
    - only condition: gather at least 150 points (50%)
    - no additional conditions on results of evaluation exam
    - *in case of Covid lockdown, 1. and 2. test change to homeworks and 3. test becomes a part of written and/or oral exam
- Final lab mark is valid only for the current academic year. You need to repeat lab work in new school year.
- **Due to Covid, grading can be changed*

Integrated Development Environment (IDE) WinIDEA

simpr - winIDEA - [C:\winIDEA\Projekti\Zgled_2020\user.s]

File View Project Simulator Debug Test Plugins Tools Window Help

Project Workspace

Filter

sample.elf

Symbols Project

user.s crt0.s sample.lcf

```
stev2:      .word 0x10
rez:        .space 4

            .align
            .global __start
__start:

            ldr r1, stev1
            ldr r2, stev2
            add r3, r2, r1
            str r3, rez

__end:      b __end
```

Memory 0x0000000

Area: Virtual Address: 0x0000000 Symbol

00000000	09	00	00	EA	08	00	00	EA
00000008	07	00	00	EA	06	00	00	EA
00000010	05	00	00	EA	04	00	00	EA
00000018	03	00	00	EA	02	00	00	EA
00000020	40	00	00	00	10	00	00	00
00000028	00	00	00	00	14	10	1F	E5
00000030	14	20	1F	E5	01	30	82	E0
00000038	18	30	0F	E5	FE	FF	FF	EA
00000040	00	00	00	00	00	00	00	00
00000048	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00
00000058	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00

Disassembly

__start

Address	Data	Disassembly
		__start
		ldr r1, stev1
000(	14101	ldr r1,[pc,-0014]
		ldr r2, stev2
000(	14201	ldr r2,[pc,-0014]
		add r3, r2, r1
000(	01308	add r3,r2,r1
		str r3, rez
000(	18300	str r3,[pc,-0018]
		__end: b __end

Registers

R0 00000000
R1 00000000
R2 00000000
R3 00000000
R4 00000000
R5 00000000
R6 00000000
R7 00000000
R8 00000000
R9 00000000

Output

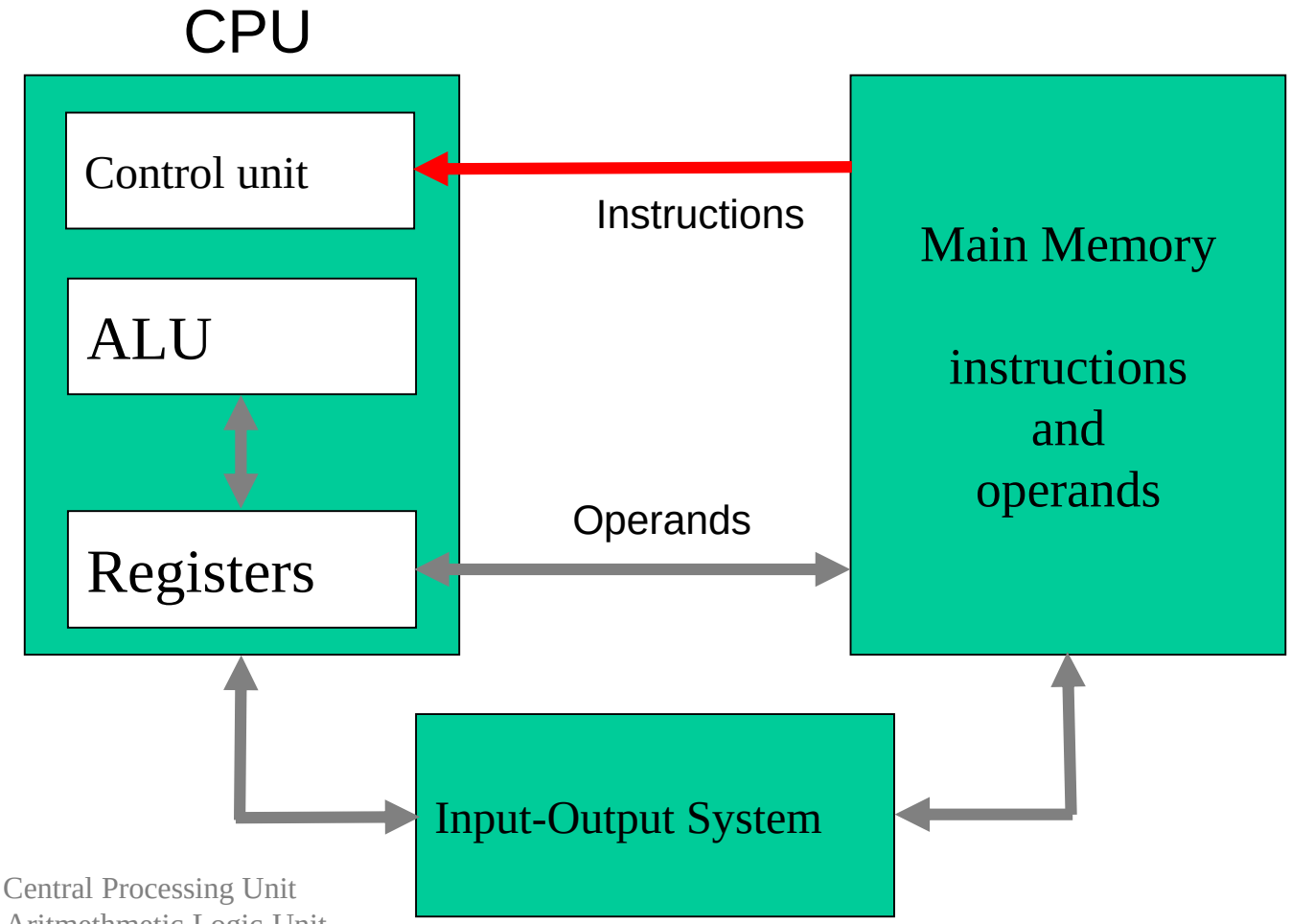
```
Compiling ...
crt0.s
user.s
Linking
"sample.elf (Dir:C:\winIDEA\Projekti\Zgled_2020\Debug\)"  ... was successfully generated.
0 Error(s)  0 Warning(s)
```

Build Find In Files Tools Script

Computer architecture CA

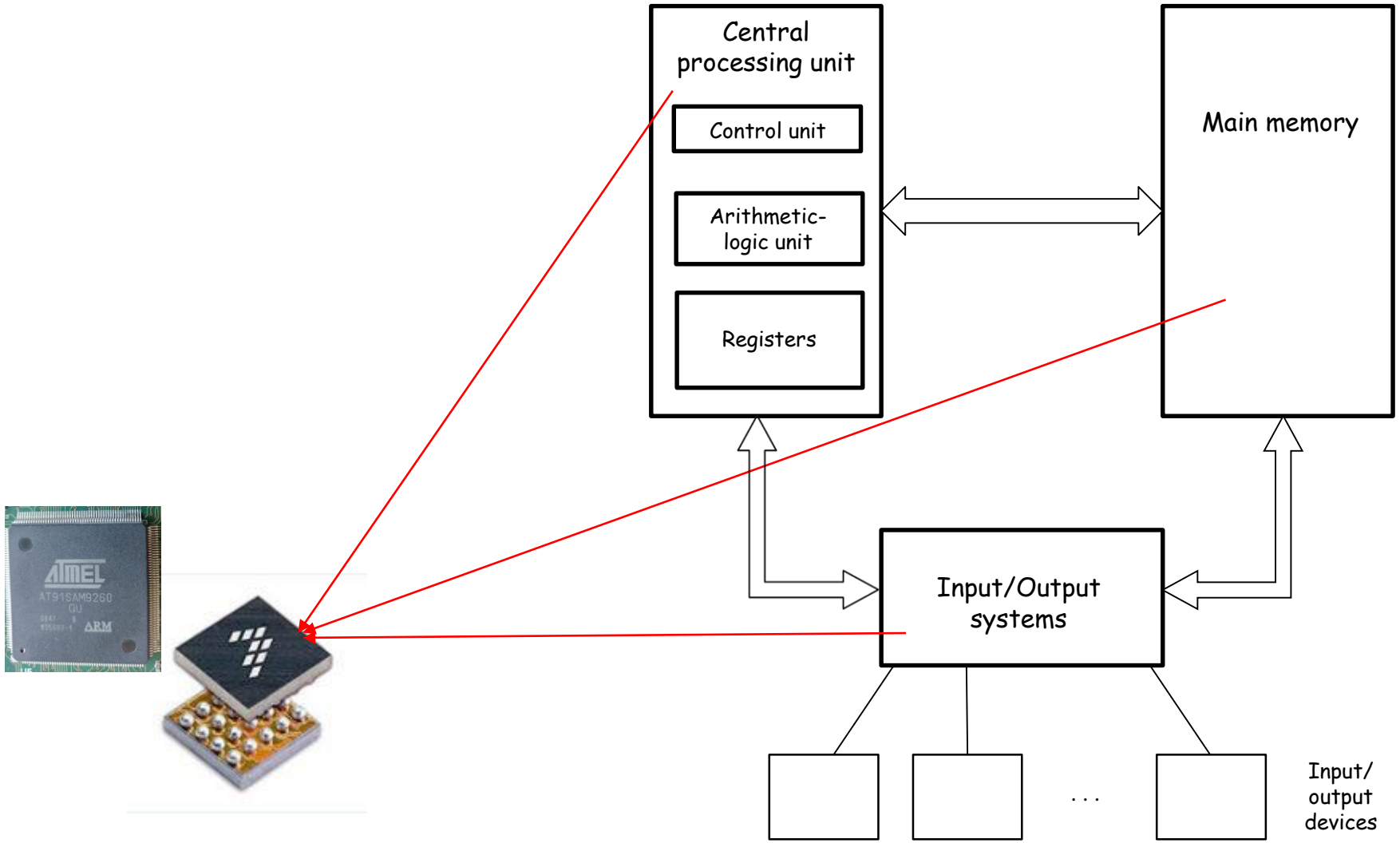
LAB 1.2 Von Neumann model (VN)

Von Neumann Computer Model



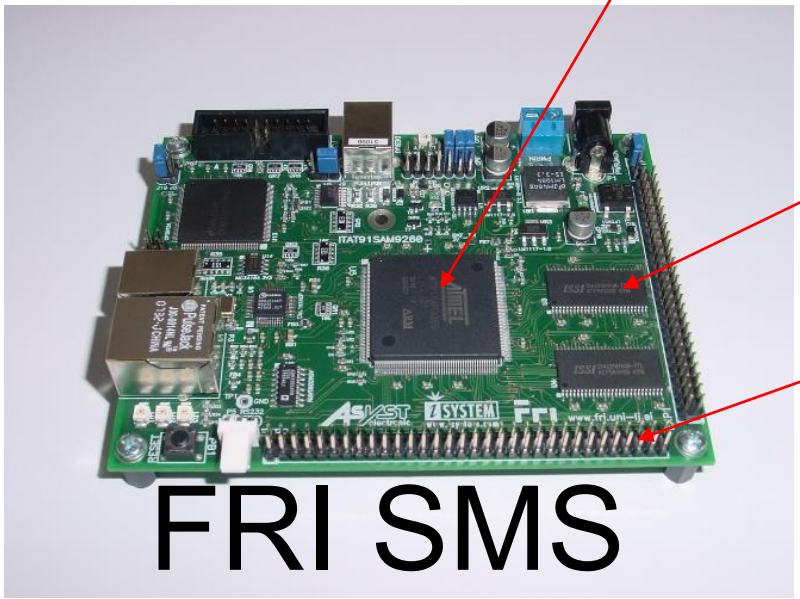
CPU – Central Processing Unit
ALU – Aritmethmetic Logic Unit

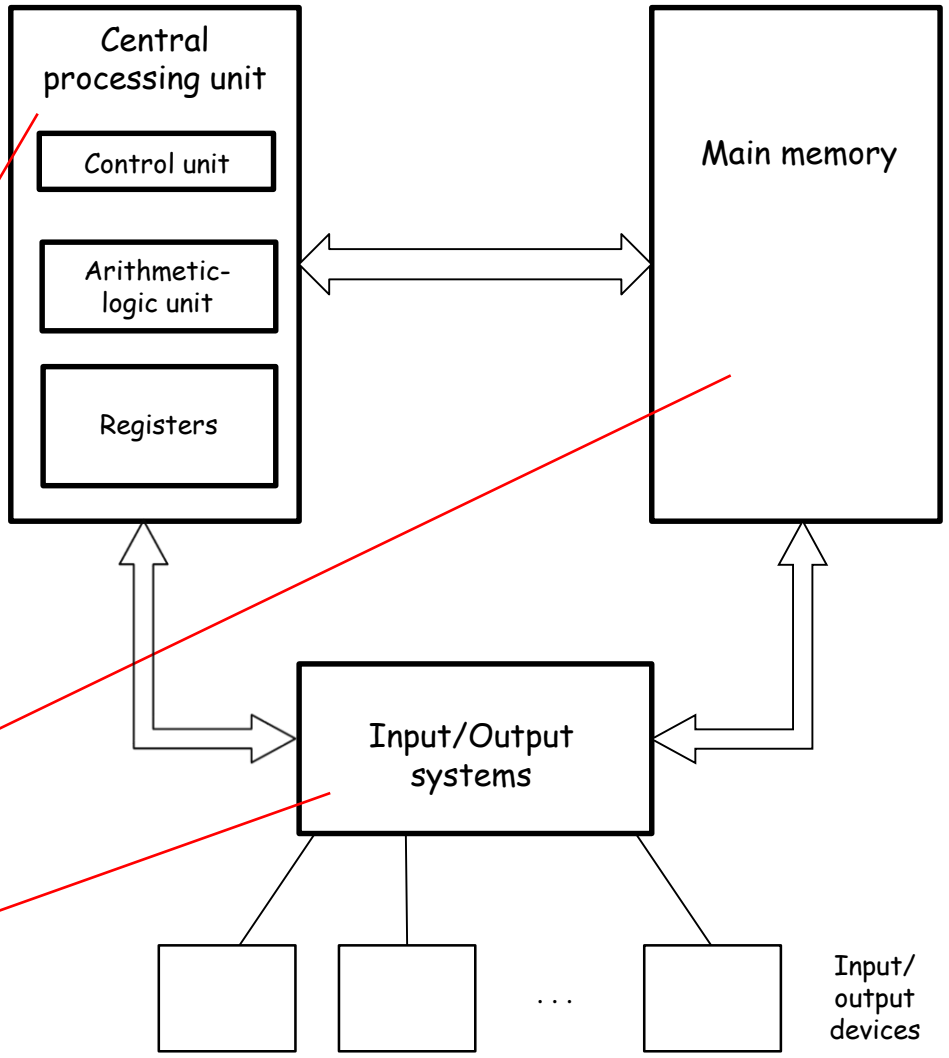
The basic computer model



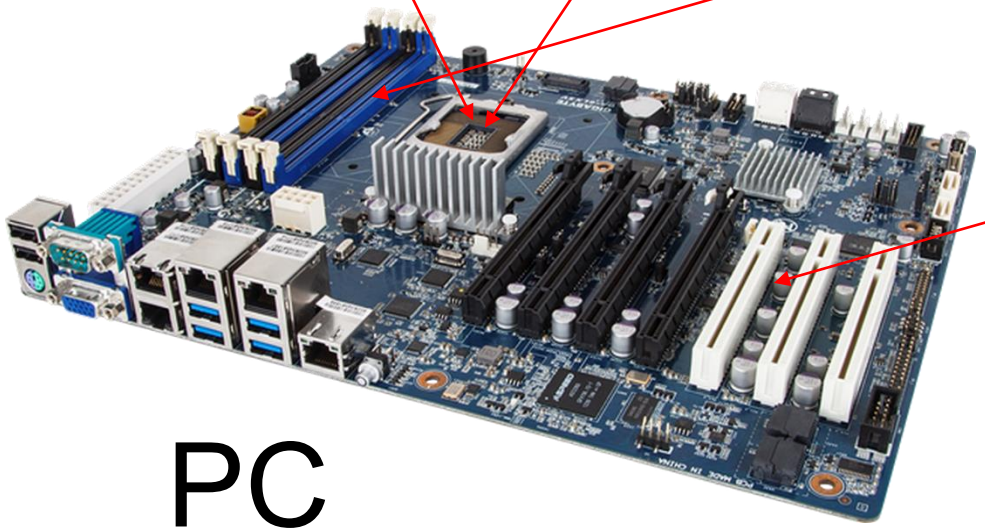
MicroControllers

The basic computer model

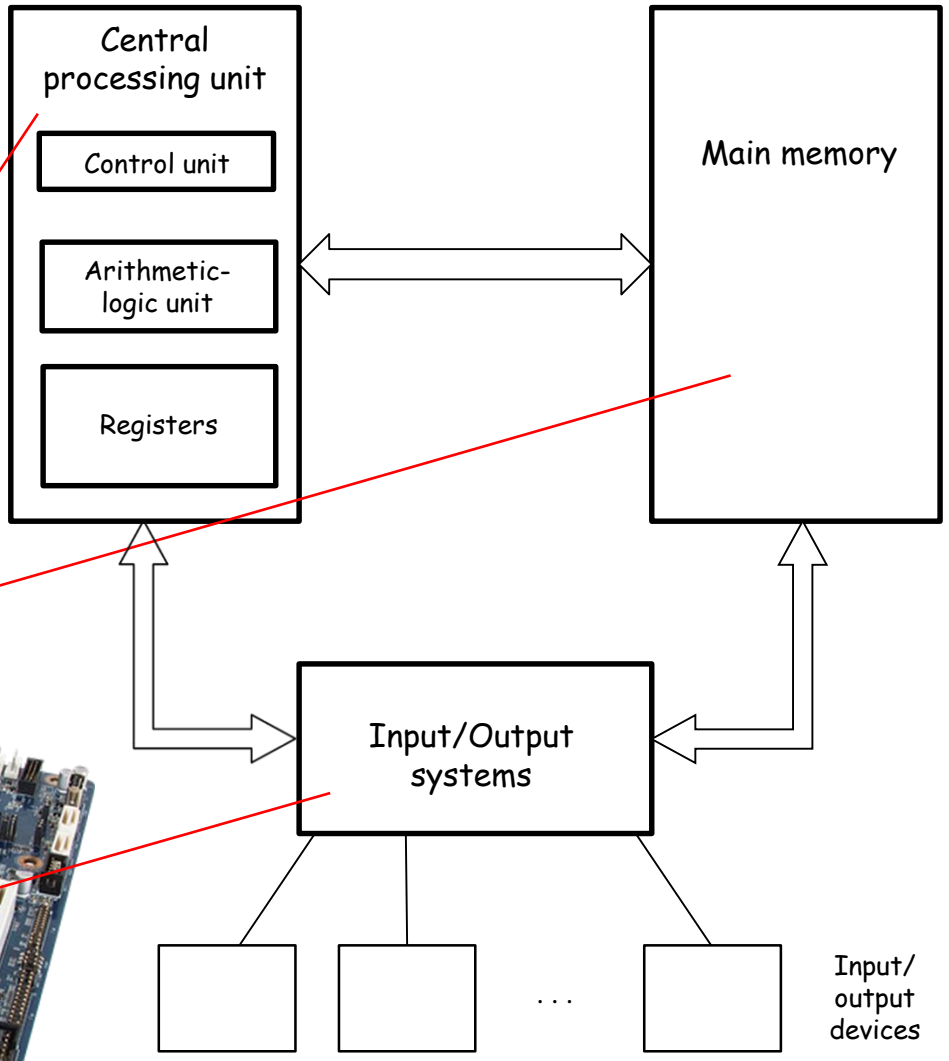




The basic computer model



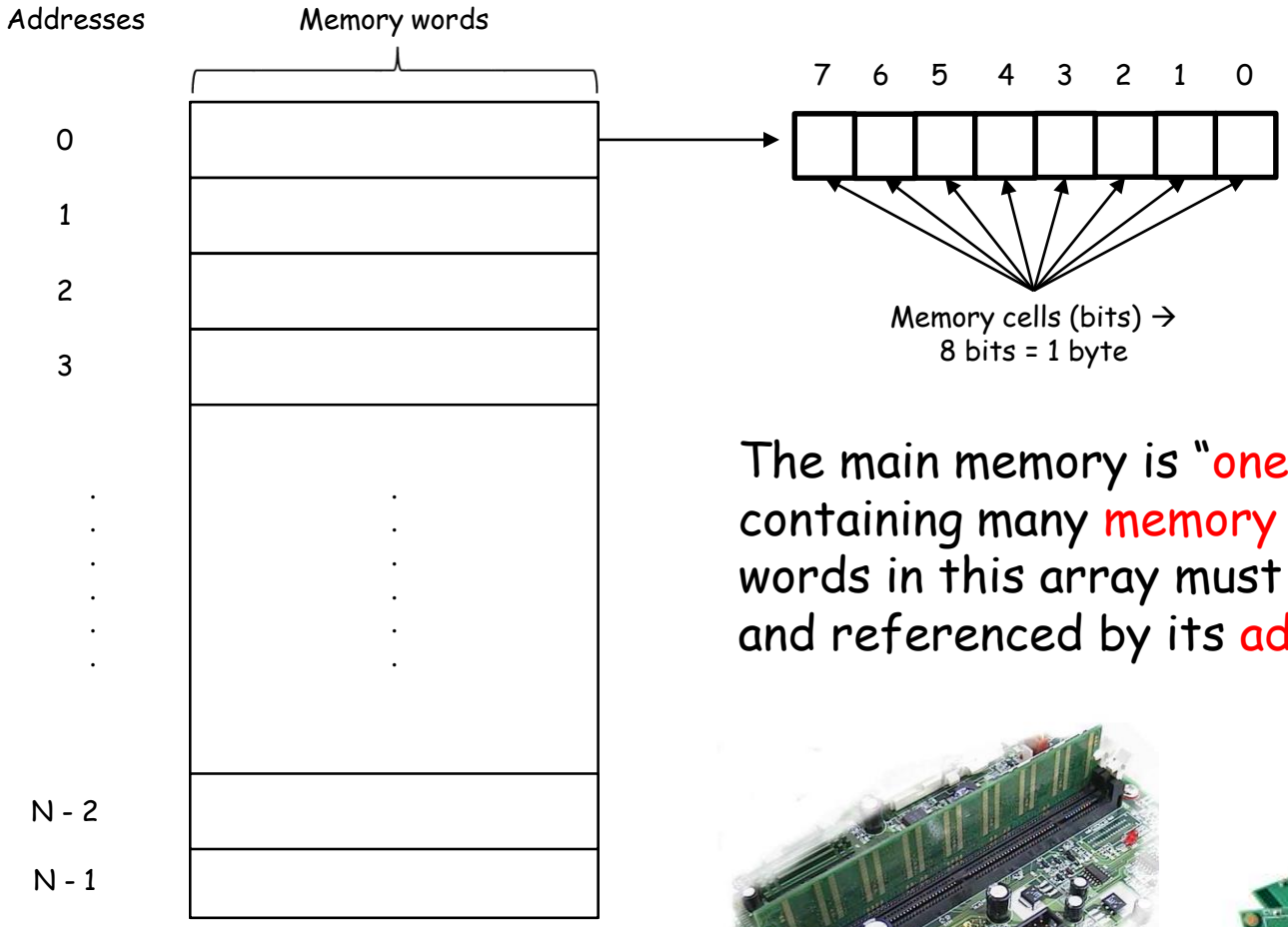
PC



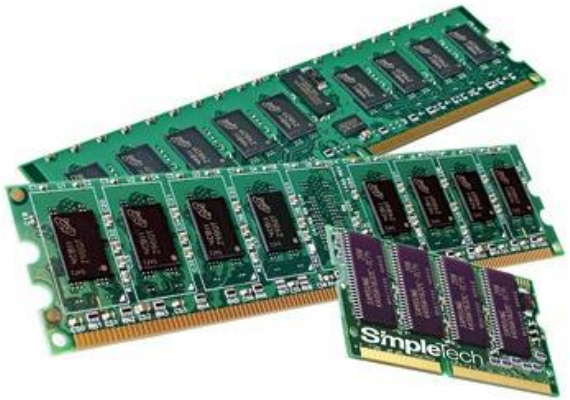
Computer architecture CA

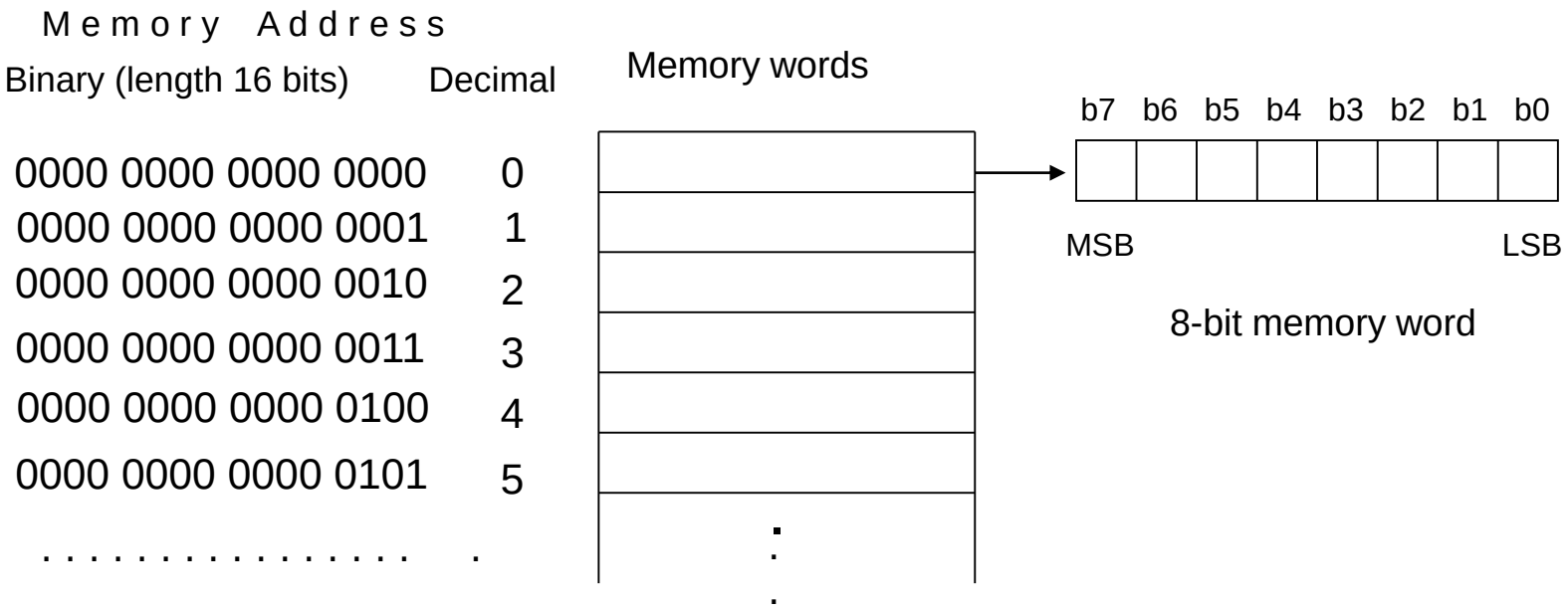
LAB 1.3 Memory

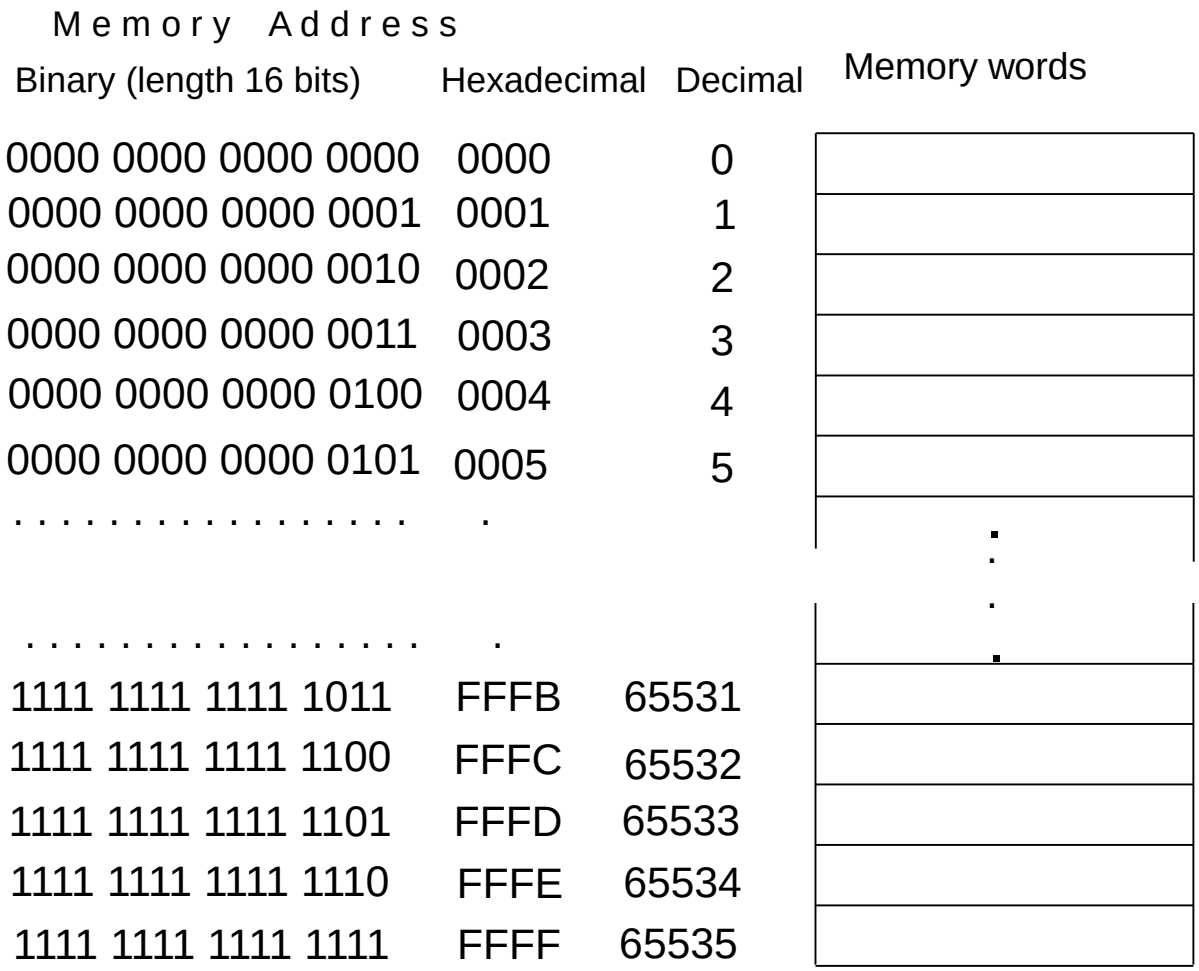
What is memory ?



The main memory is "one dimensional array" containing many memory words. Each memory words in this array must be uniquely identified and referenced by its address!



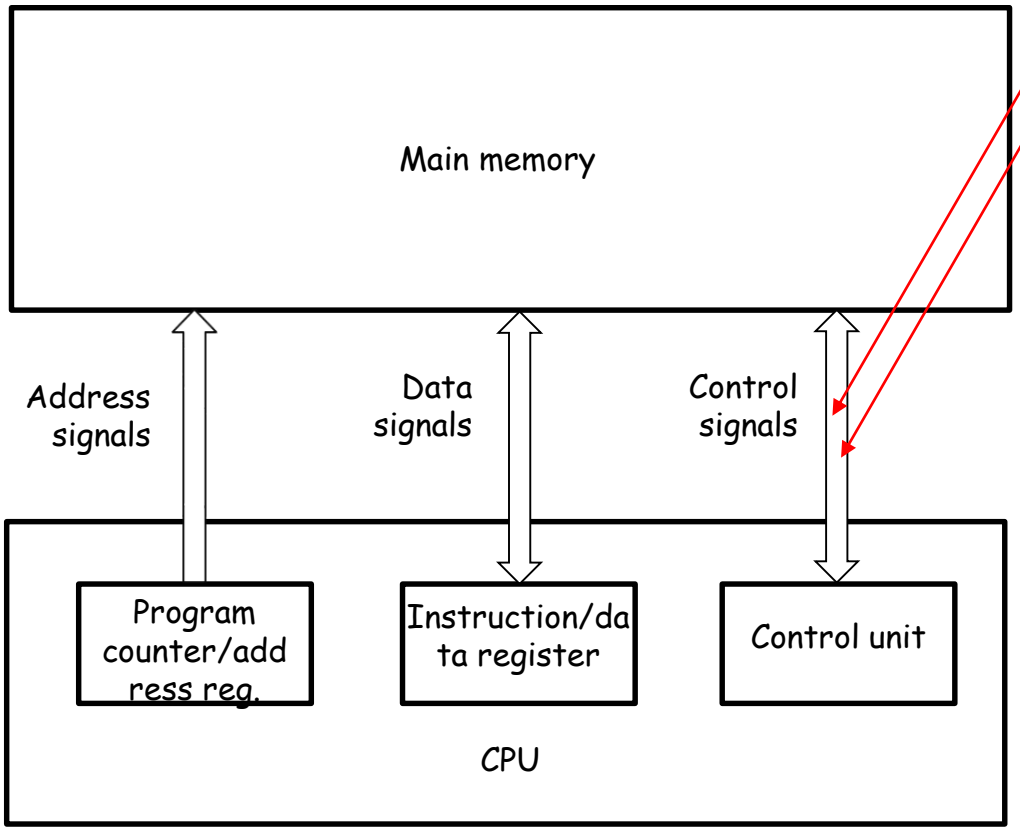

Memory Address
Binary (length 16 bits)
Decimal
0000 0000 0000 0000
0
0000 0000 0000 0001
1
0000 0000 0000 0010
2
0000 0000 0000 0011
3
0000 0000 0000 0100
4
0000 0000 0000 0101
5
Memory words
b7 b6 b5 b4 b3 b2 b1 b0
MSB
LSB
8-bit memory word


Memory Address
Binary (length 16 bits)
Hexadecimal
Decimal
Memory words
0000 0000 0000 0000 0000 0
0000 0000 0000 0001 0001 1
0000 0000 0000 0010 0002 2
0000 0000 0000 0011 0003 3
0000 0000 0000 0100 0004 4
0000 0000 0000 0101 0005 5
1111 1111 1111 1011 FFFB 65531
1111 1111 1111 1100 FFFC 65532
1111 1111 1111 1101 FFFD 65533
1111 1111 1111 1110 FFFE 65534
1111 1111 1111 1111 FFFF 65535

Interconnection CPU <-> main memory

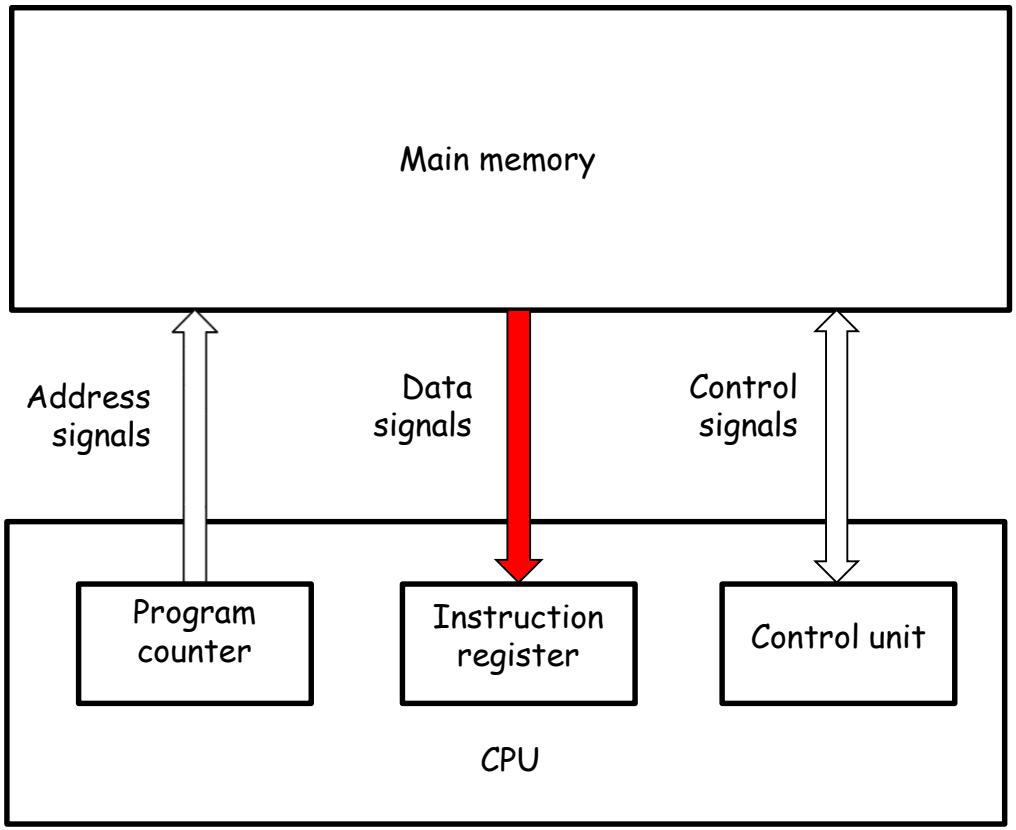
Bus = group of lines
(Address, Data,
Control buses)

Line = physical connection
Signal = content transferred over the line (1bit)



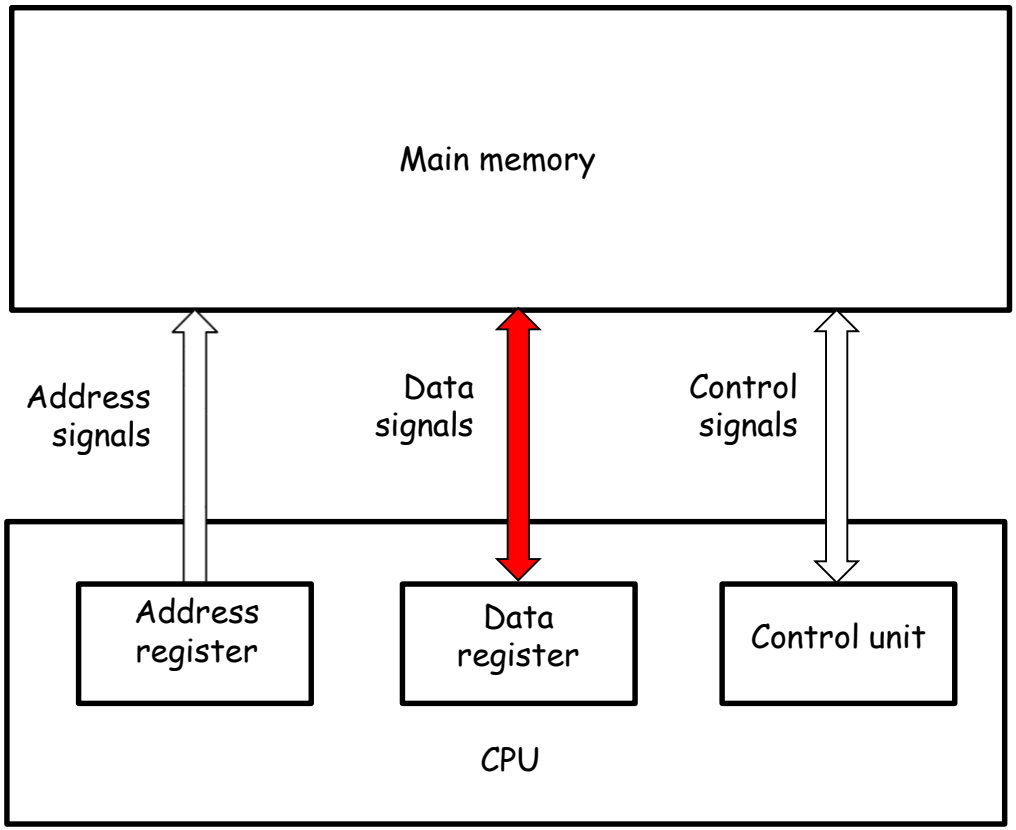
How does CPU access the main memory?

Example for accessing instructions:

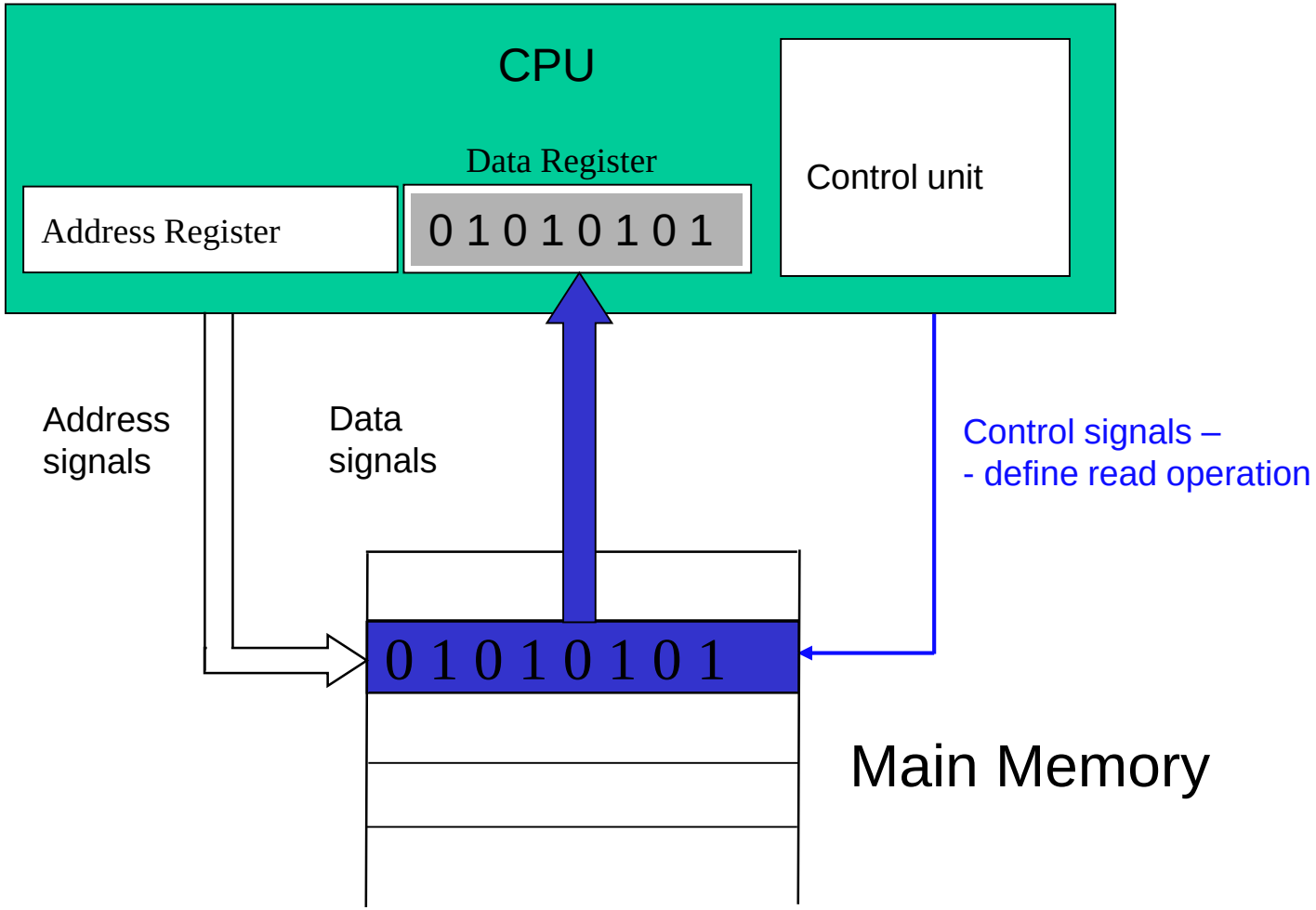


How does CPU access the main memory?

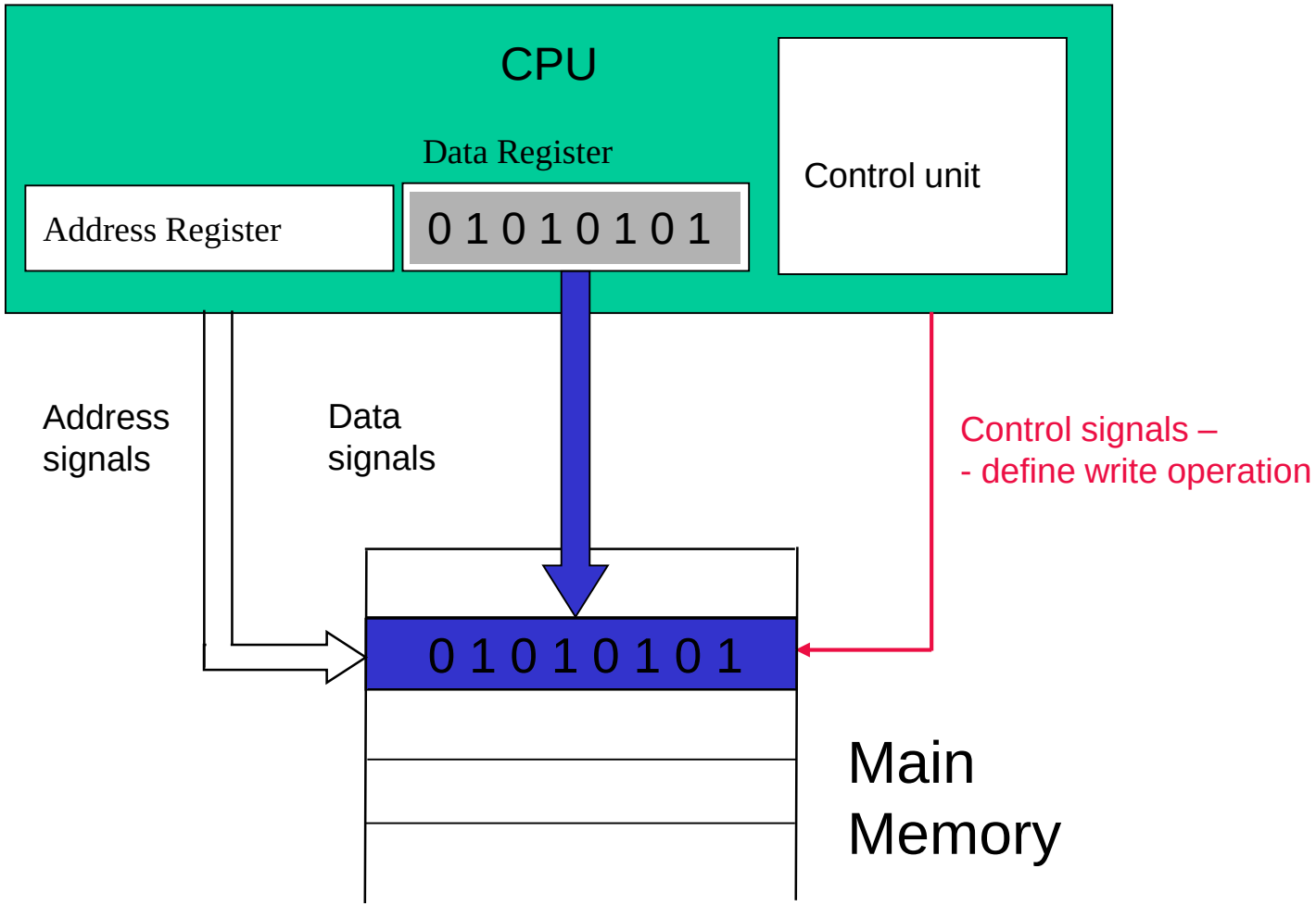
Examples for accessing operands:



CPU and Main Memory – read access



CPU and Main Memory – write access



Computer architecture CA

LAB 1.4 Quick intro to numeral systems

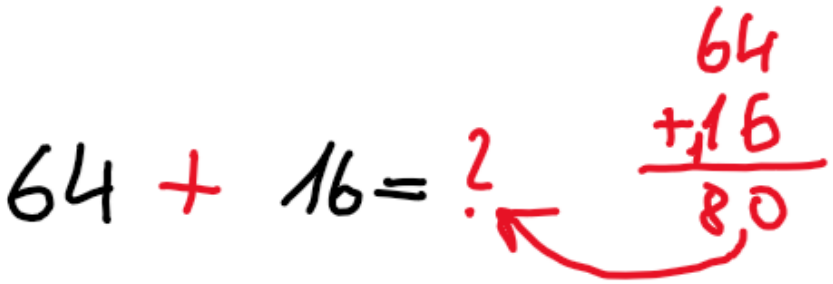
Computer architecture CA

LAB 1.5 Big and Little Endian rules

Computer architecture CA

LAB 1.6 Addition – human, python, assembler cases

Human (case: 64 + 16 = 80)



Python (case: REZ = STEV1 + STEV2)

Adding two variables in Python.

http://goo.gl/YXQ5qN

Python 2.7

```
1  STEV1=0x40
2  STEV2=0x10
3  REZ = STEV1 + STEV2
4  print (" STEV1 = " + hex(STEV1) + "\n+STEV2 = " + hex(STE
```

Frames Objects

Global frame

STEV1	64
STEV2	16
REZ	80

Print output (drag lower right corner to resize)

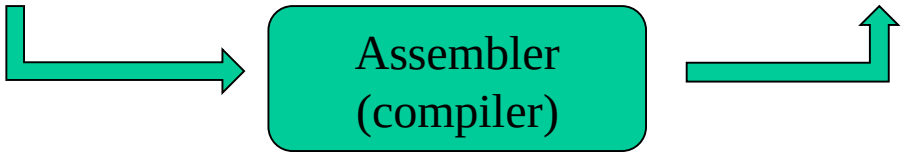
```
 STEV1 = 0x40
+STEV2 = 0x10
-------------
   REZ = 0x50
```

WinIDEA (case: rez = stev1 + stev2)

Evaluate the sum of two variables in ARM assembler. Download prepared project from e-classroom.

Variables values are stored in the main memory. We perform a simple arithmetic addition with the following instructions:

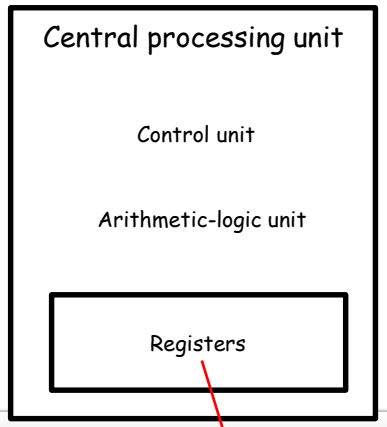
Assembly language	Instruction description	Machine language
ldr r1, stev1	R1 ← M[0x20]	0xE51F1014
ldr r2, stev2	R2 ← M[0x24]	0xE51F2014
add r3, r2, r1	R3 ← R1 + R2	0xE0823001
str r3, rez	M[0x28] ← R3	0xE50F3018

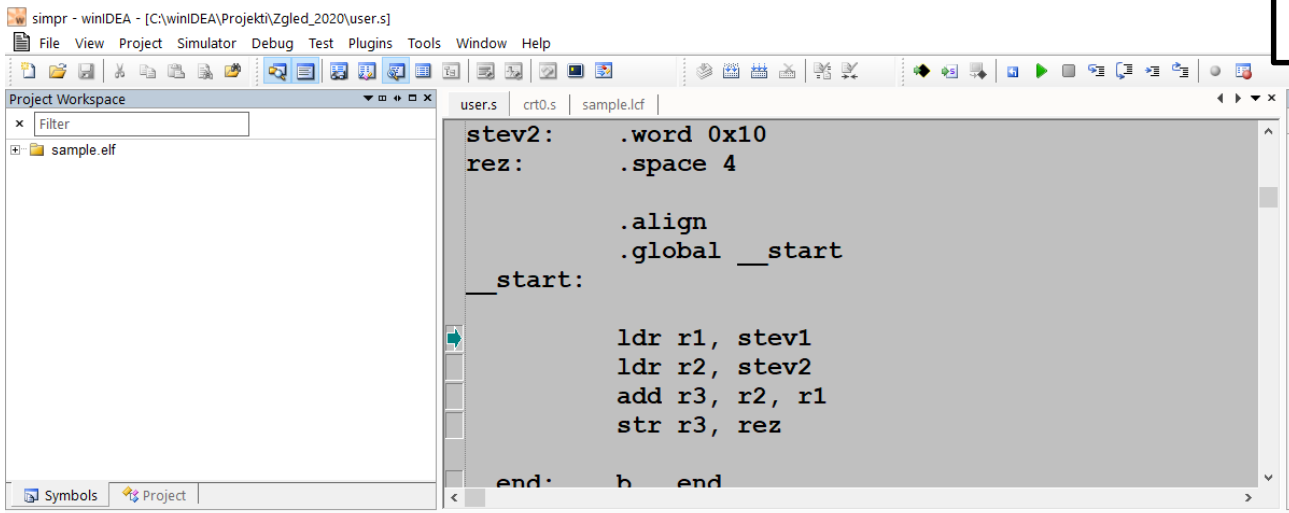


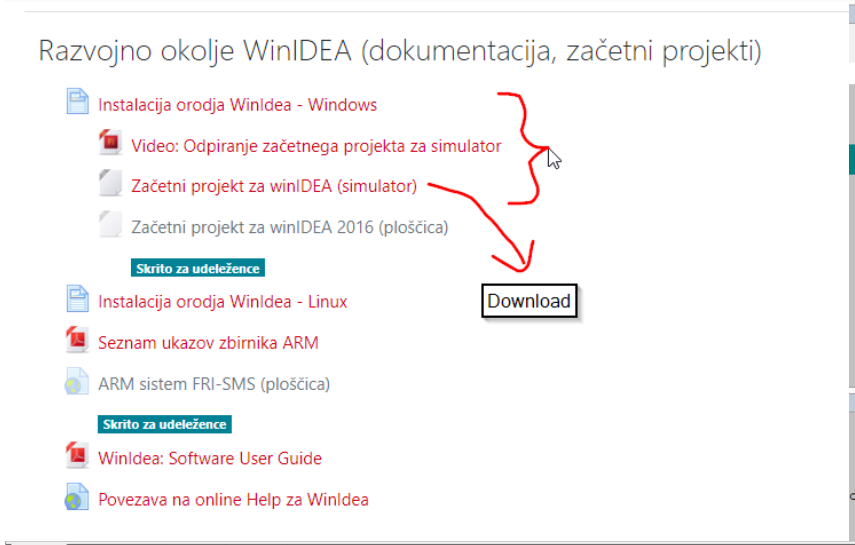
Execute instructions step-by-step and observe the register's values and the variable's values inside the main memory.

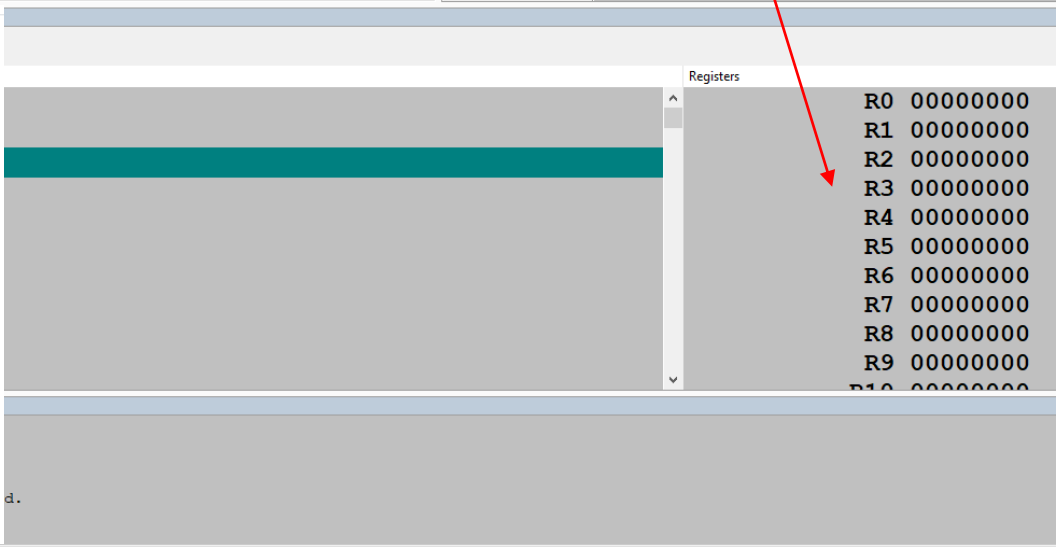
Practical example „zgled“

Integrated Development Environment (IDE) WinIDEA









Computer architecture CA

LAB 1.7 Notes templates

Python (case: REZ = STEV1 + STEV2)

Frames Objects

Global frame

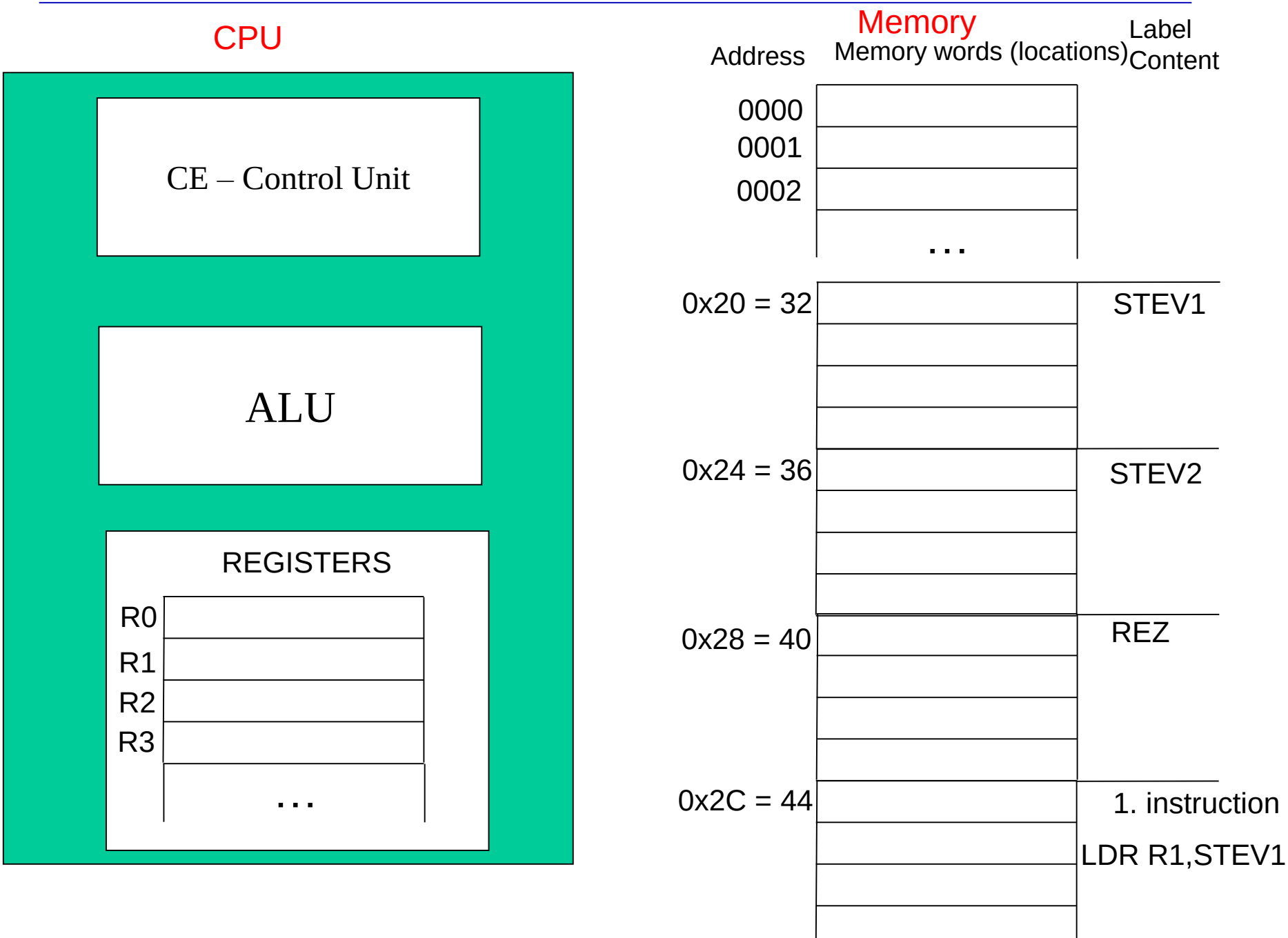
STEV1	64
STEV2	16
REZ	80

Python 2.7

```
1  STEV1=0x40
2  STEV2=0x10
3  REZ = STEV1 + STEV2
4  print (" STEV1 = " + hex(STEV1) + "\n+STEV2 = " +
```

http://goo.gl/YXQ5qN

Zgled: adding two numbers



Zgled: adding two numbers

INSTRUCTIONS

	Machine Instr.	Assembly Instr.	Description	Comment
1.	0xE51F1014	ldr r1, stev1	R1 ← M[0x20]	
2.	0xE51F2014	ldr r2, stev2	R2 ← M[0x24]	
3.	0xE0823001	add r3, r2, r1	R3 ← R1 + R2	
4.	0xE50F3018	str r3, rez	M[0x28] ← R3	

Pravilo tankega in debelega konca / Big vs. Little Endian

MSB LSB

0 x AA BB CC DD

Debeli konec
Big Endian

Tanki konec
Little Endian