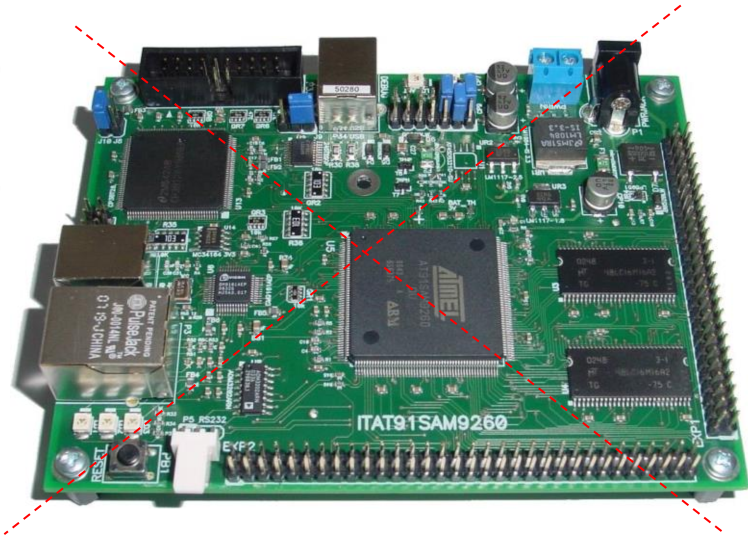
Computer architecture CA

Computer STM32H750-DK

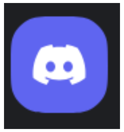
- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9

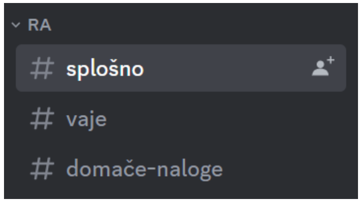


Team CA

Tutors

Žiga Pušnik
ziga.pusnik@fri....





https://discord.com/channels/1108375680997130271/1156262933416902757

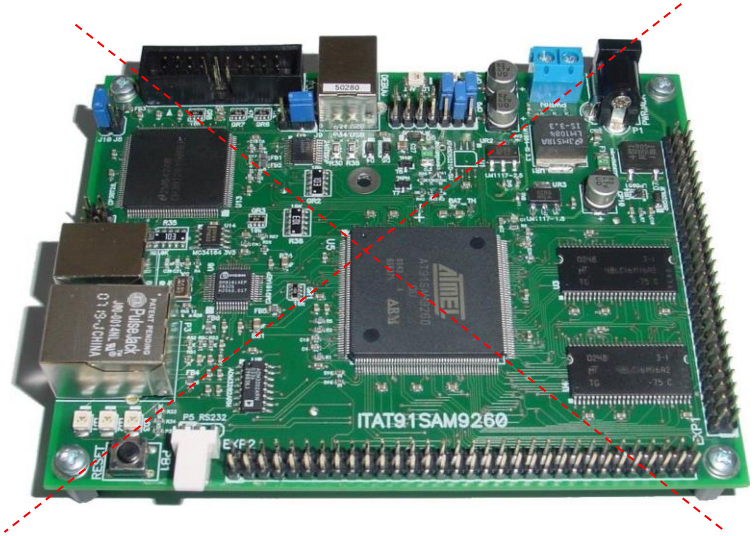
Andrej Sušnik
as1767@student.uni-lj.si

Robert Rozman
rozman@fri.uni-lj.si

Computer architecture CA

Computer STM32H750-DK

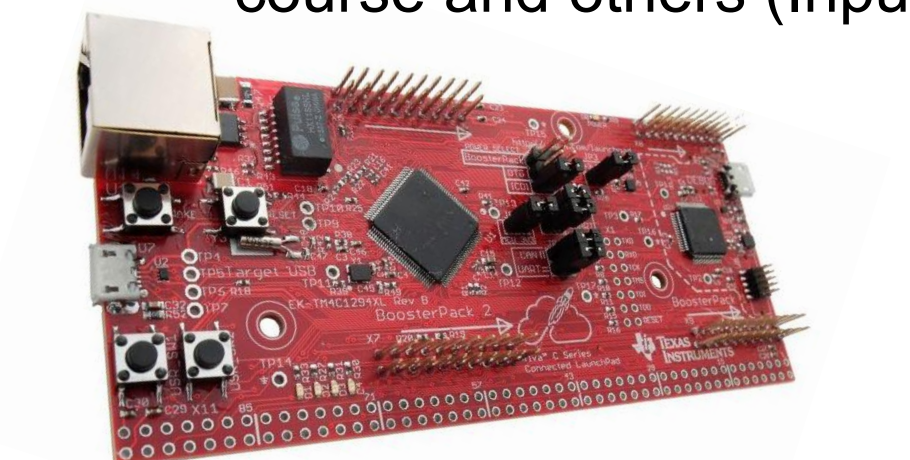
- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9



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 (Input/Output devices, …)



Content of LAB work

- Basic knowledge needed from lectures (e.g. memory address, memory words, …)
- Core: Programming in ARM assembly language
- Format:
  - lab exercises (2 hands-on 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

Evaluation – grading*

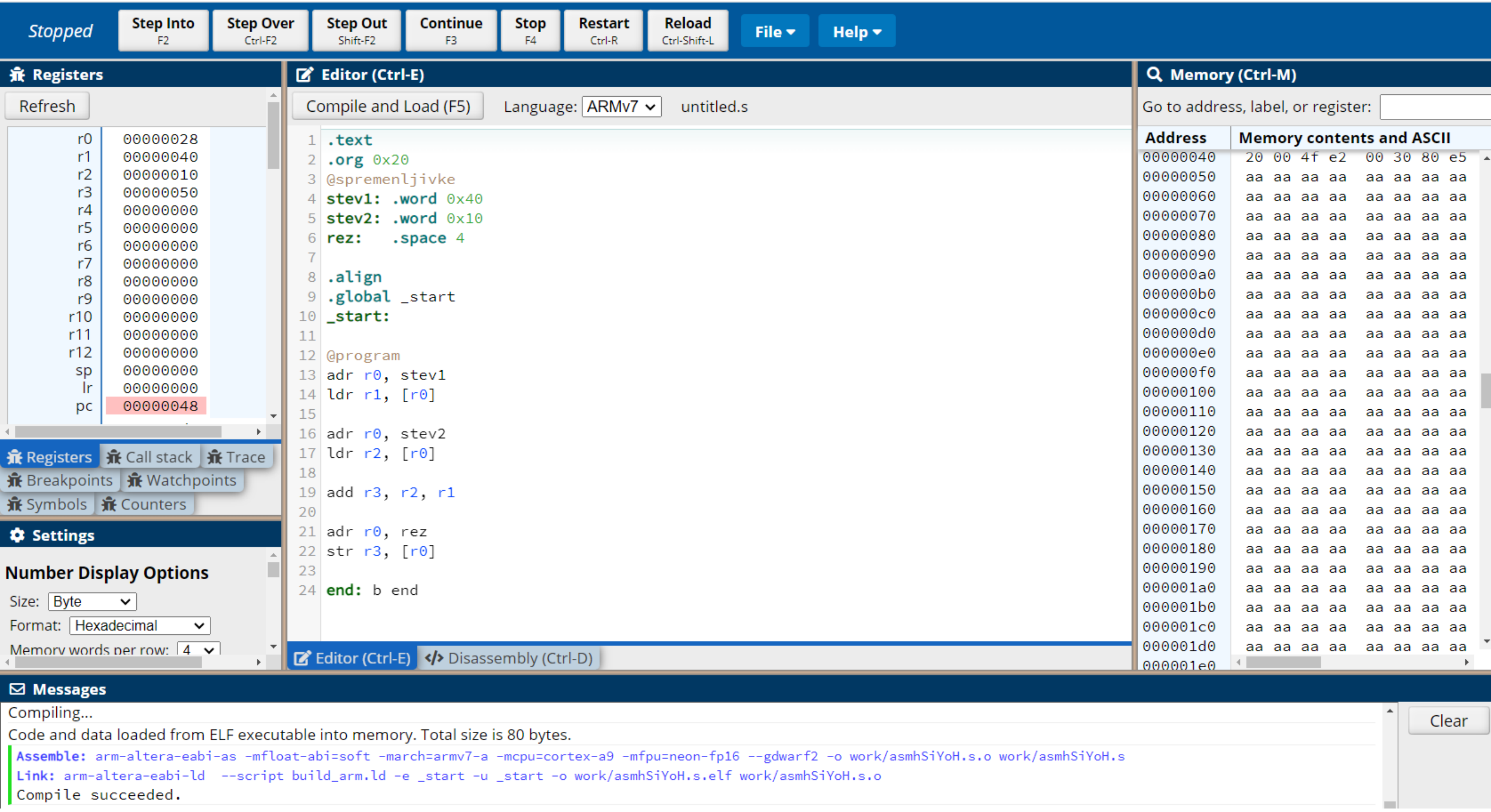
- Lab marks represents 50% of the final mark for the course. You need to have:
  - successfully evaluated lab work (presence, work)
  - successfully evaluated homework assignment,
  - three intermediate evaluation exams (80 + 100 + 120 points)
    - only condition: gather at least 150 points (50%)
    - 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 grade 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*

Web simulator cpulator

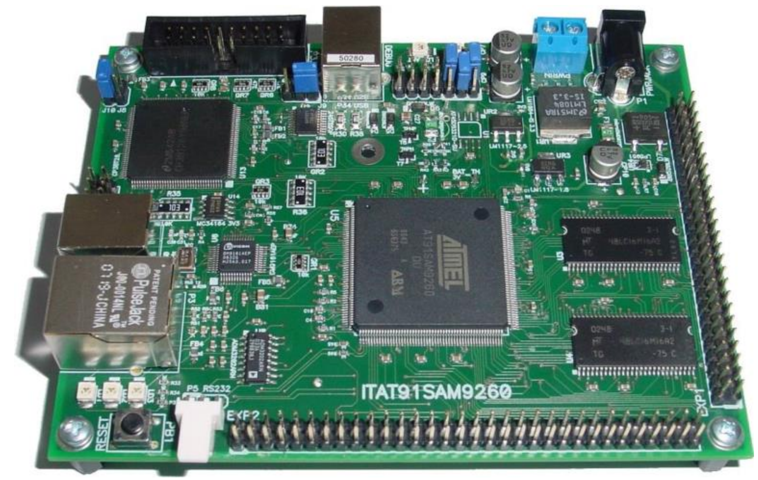
- https://cpulator.01xz.net/?sys=arm
- Base project for CA course:
  - https://cpulator.01xz.net/?sys=arm&loadasm=share/sg8LlNt.s



Computer architecture CA

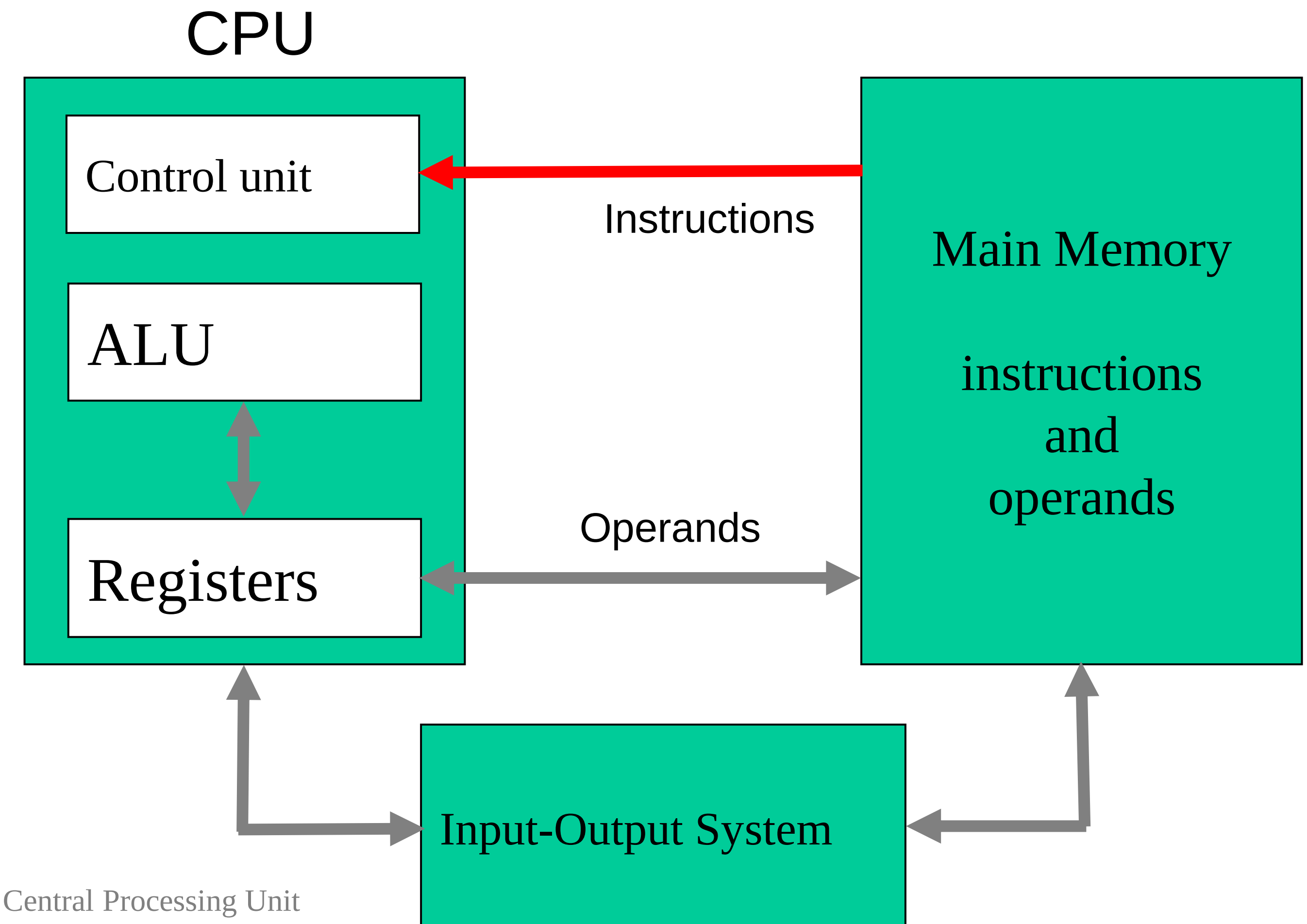
Computer STM32H750-DK

- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9



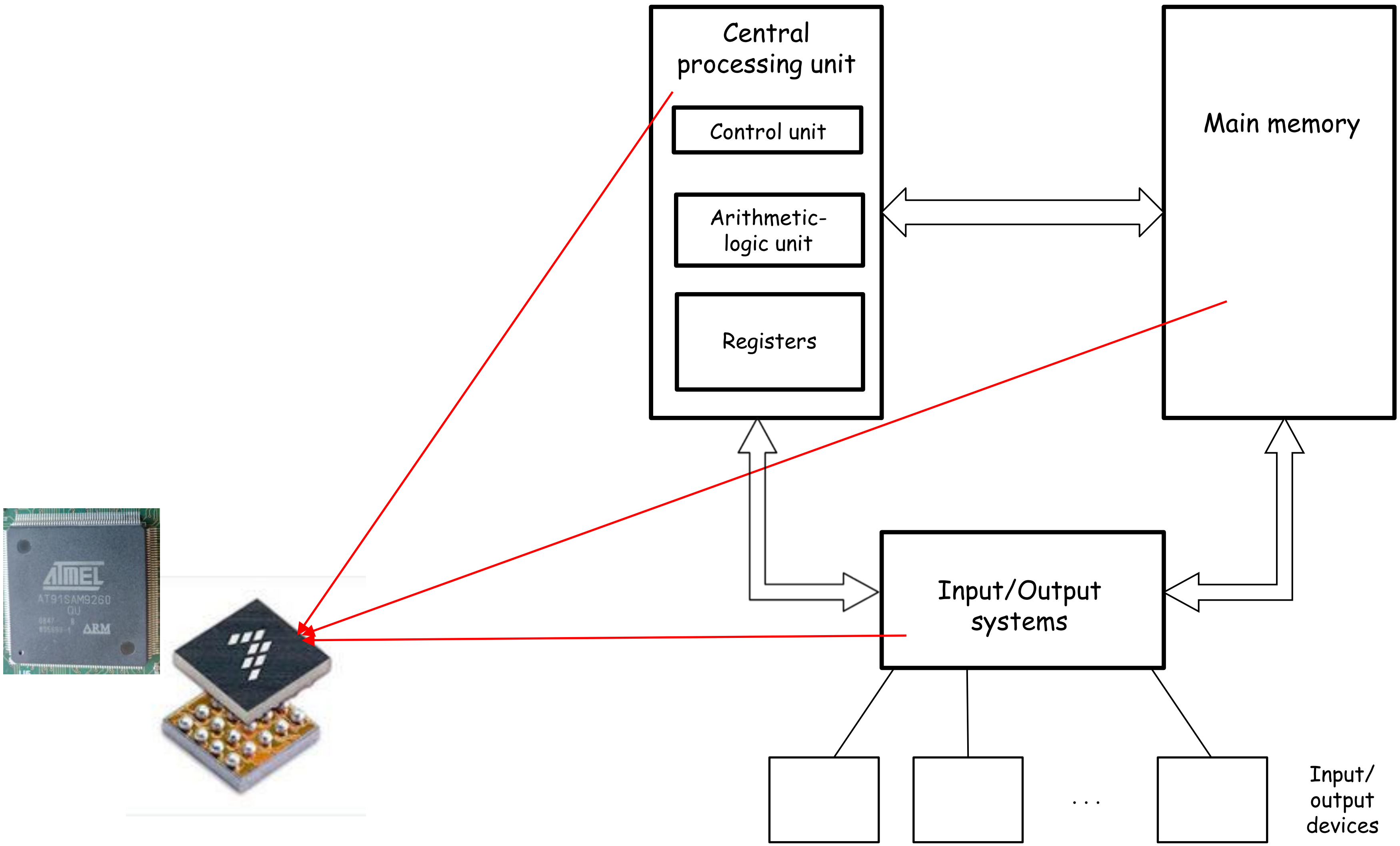
LAB 1.2 Von Neumann model (VN)

Von Neumann Computer Model



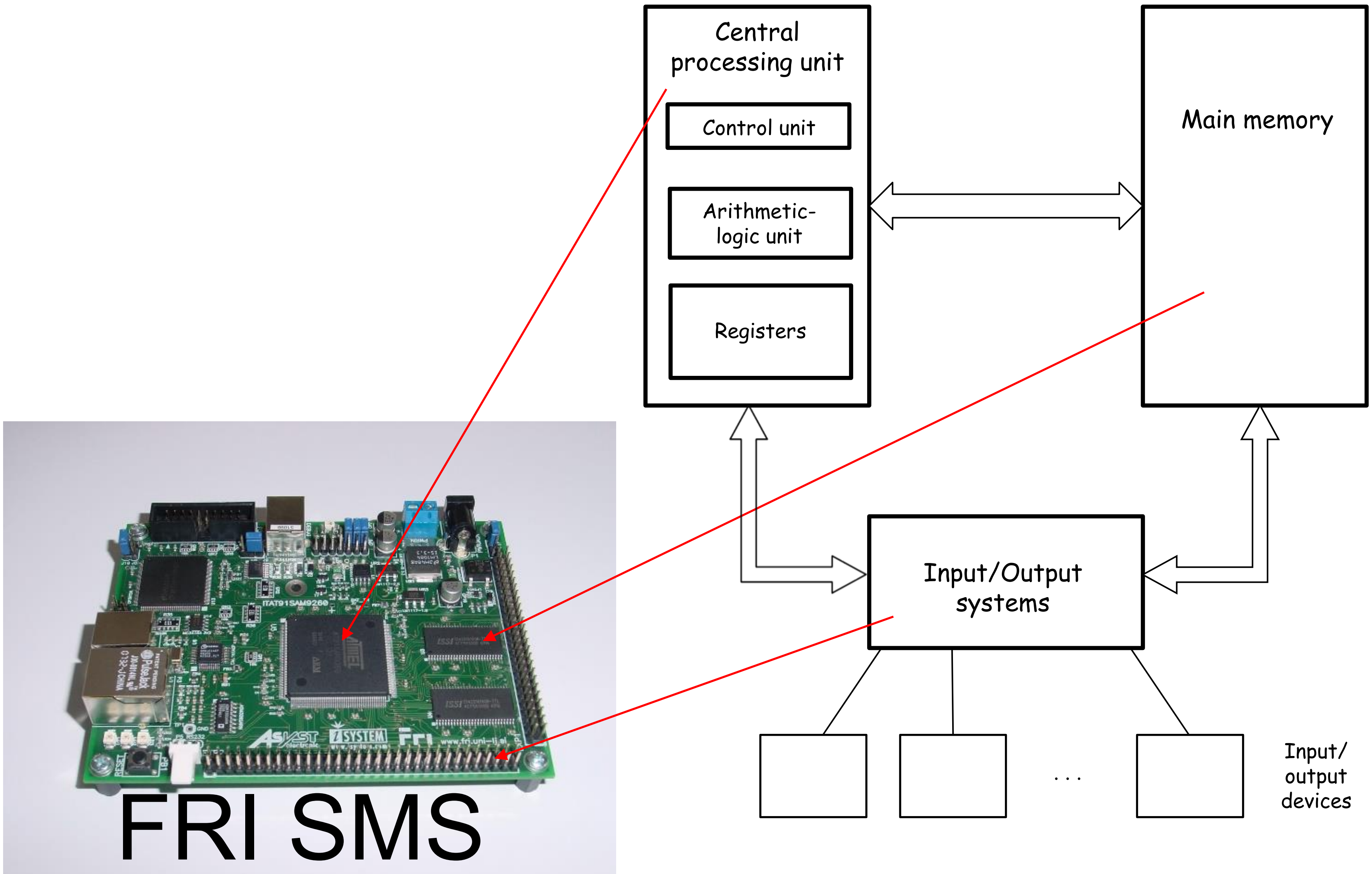
CPU – Central Processing Unit
ALU – Aritmethmetic Logic Unit

The basic computer model

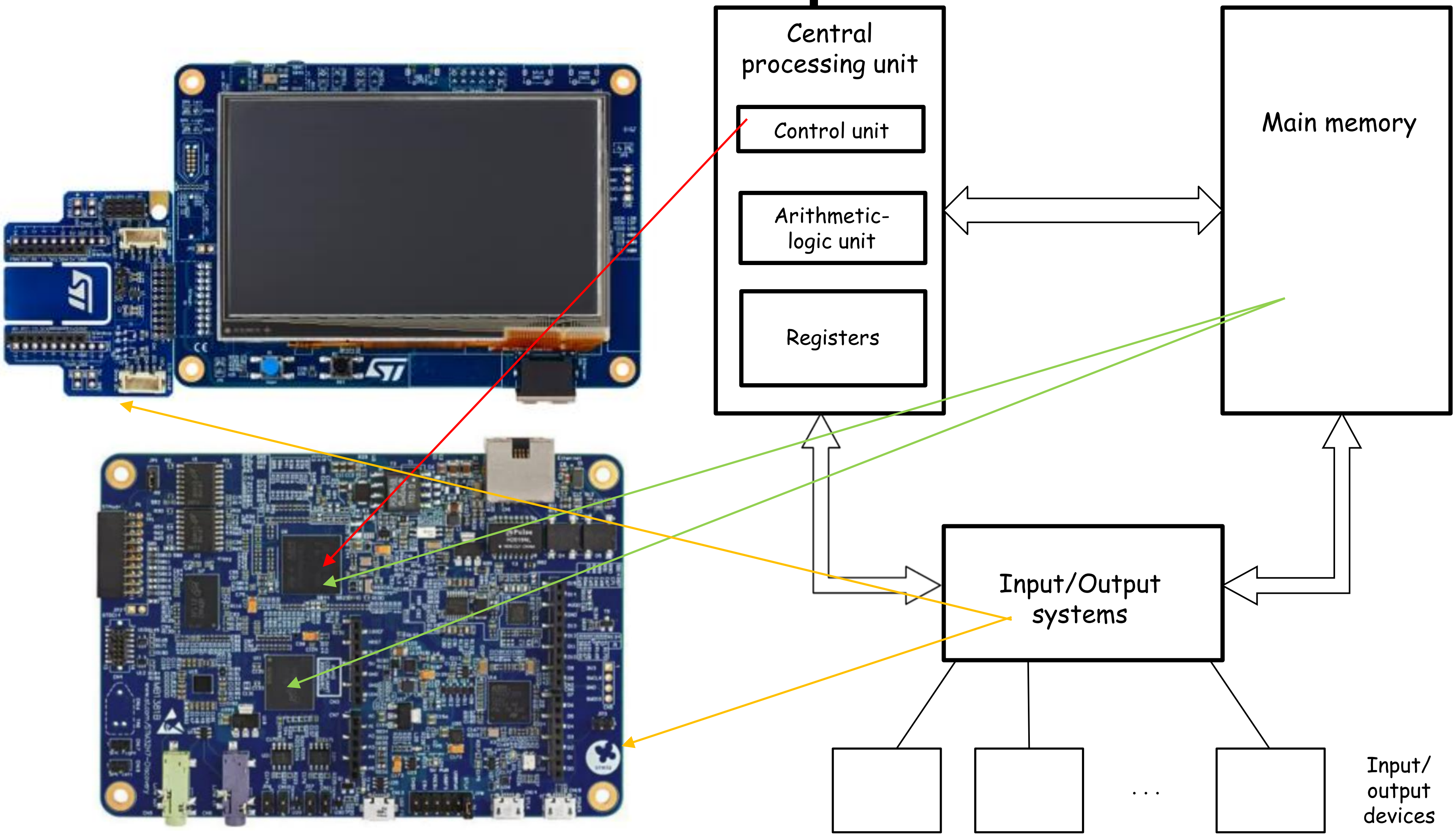


MicroControllers

The basic computer model

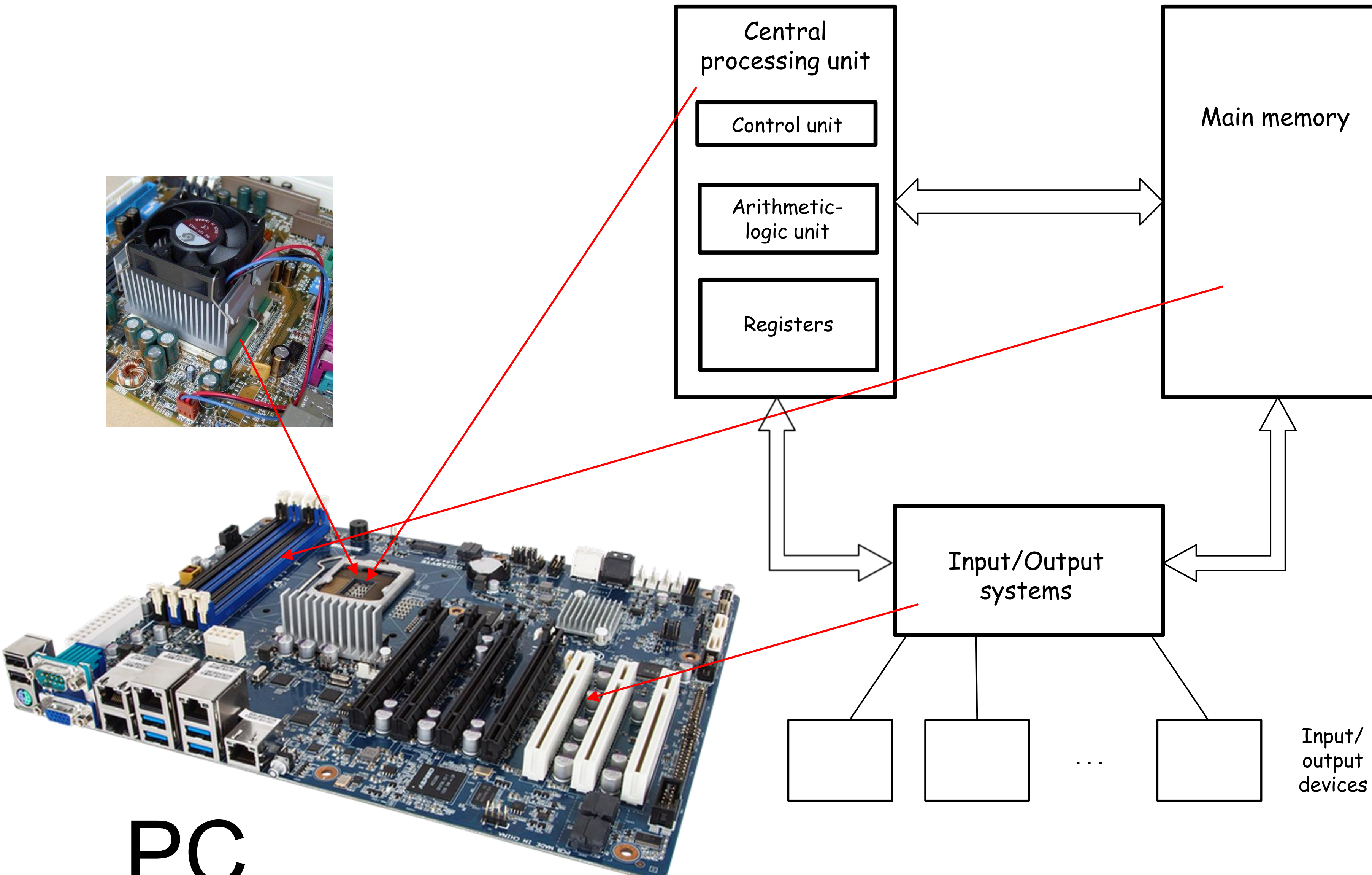


The basic computer model



STM32H750-DK

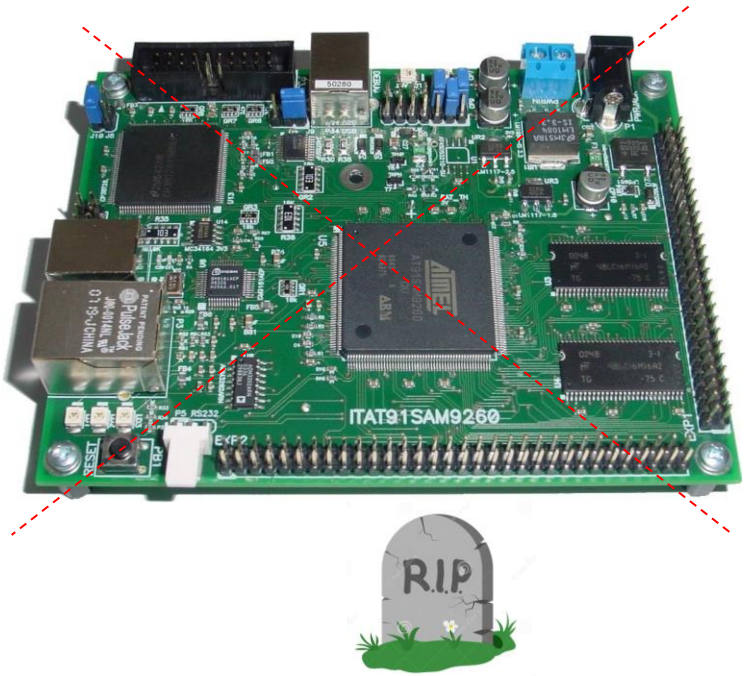
The basic computer model



Computer architecture CA

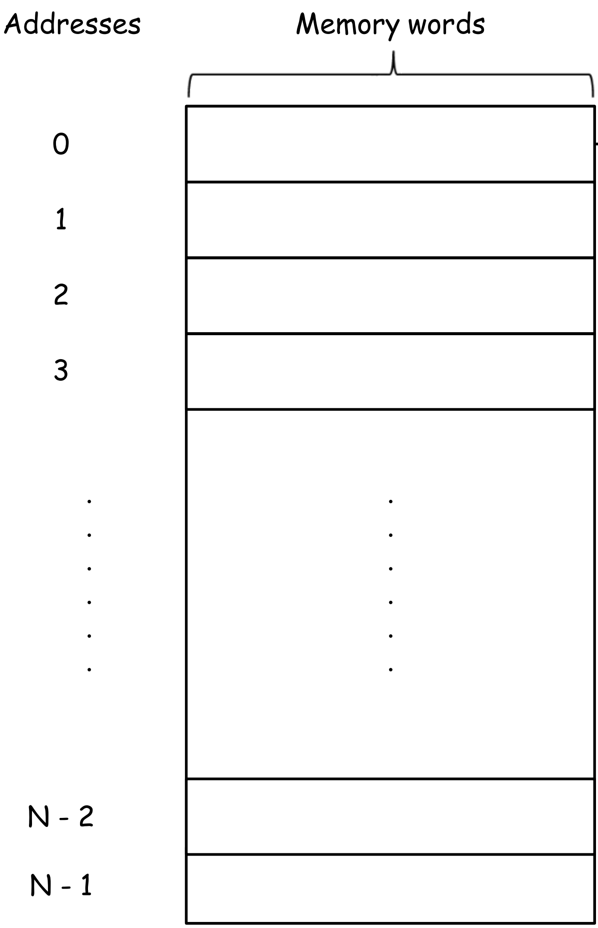
Computer STM32H750-DK

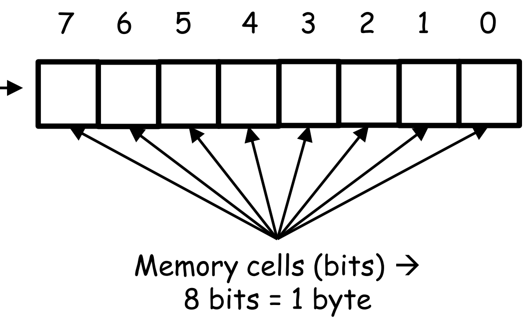
- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9



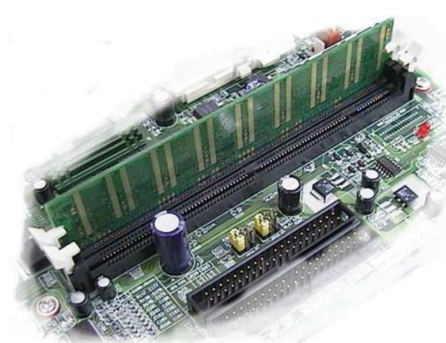
LAB 1.3 Memory

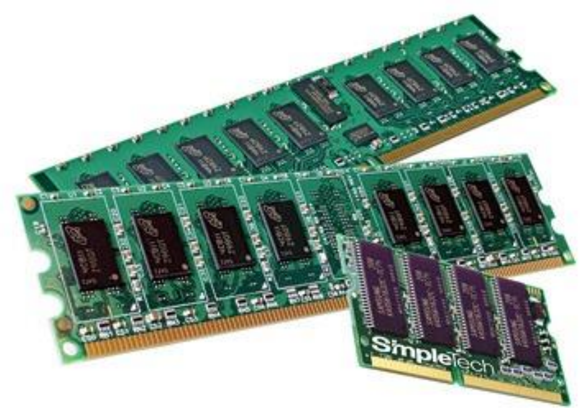
What is memory ?





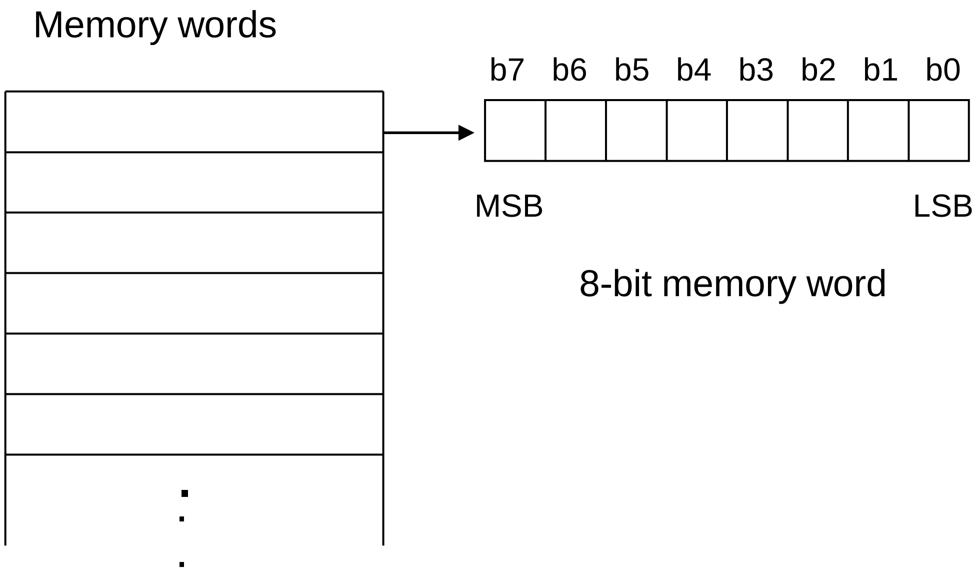
The main memory is "one dimensional array" containing many memory words. Each memory words in this array must be uniquely identified and referenced by its address!





Main Memory in von Neumann Computer

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
. . . .	.



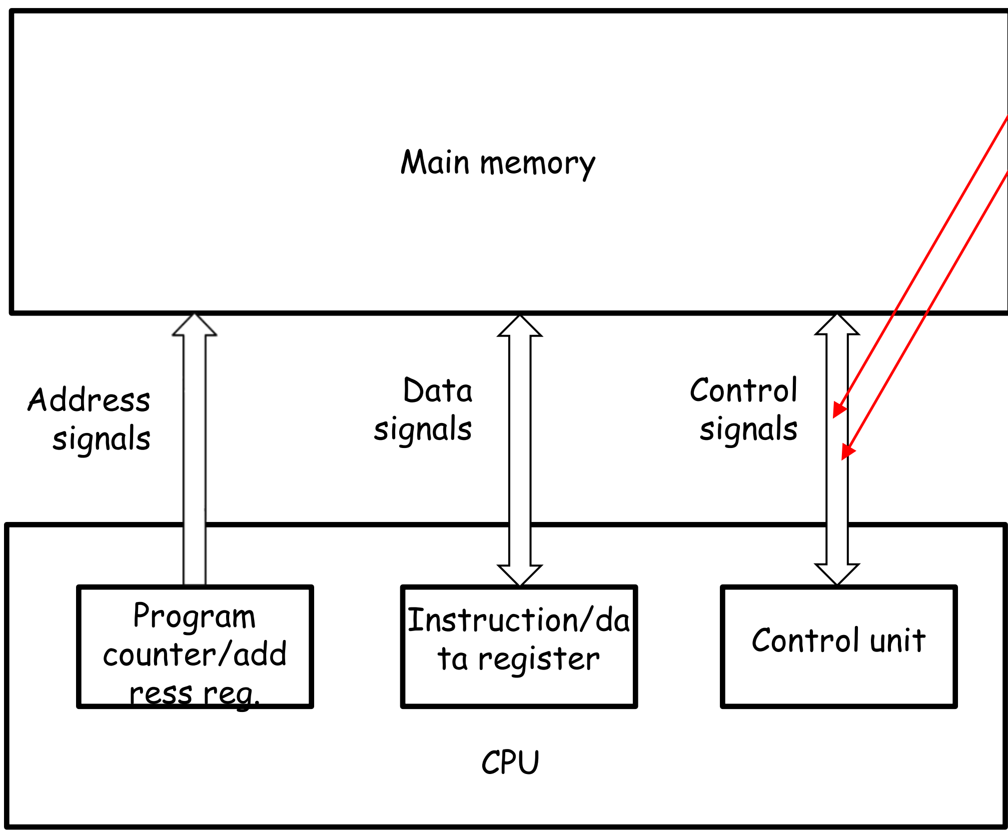
Memory Address

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

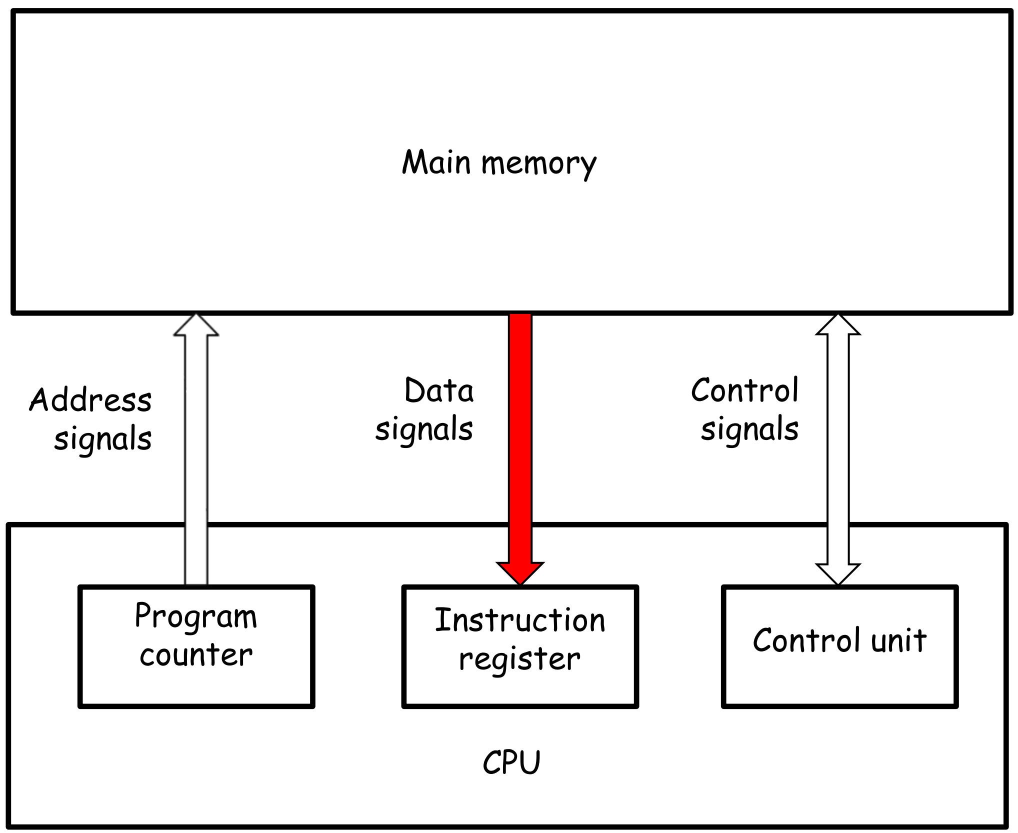
Bus = a group of related lines
(Address, Data, Control buses)

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



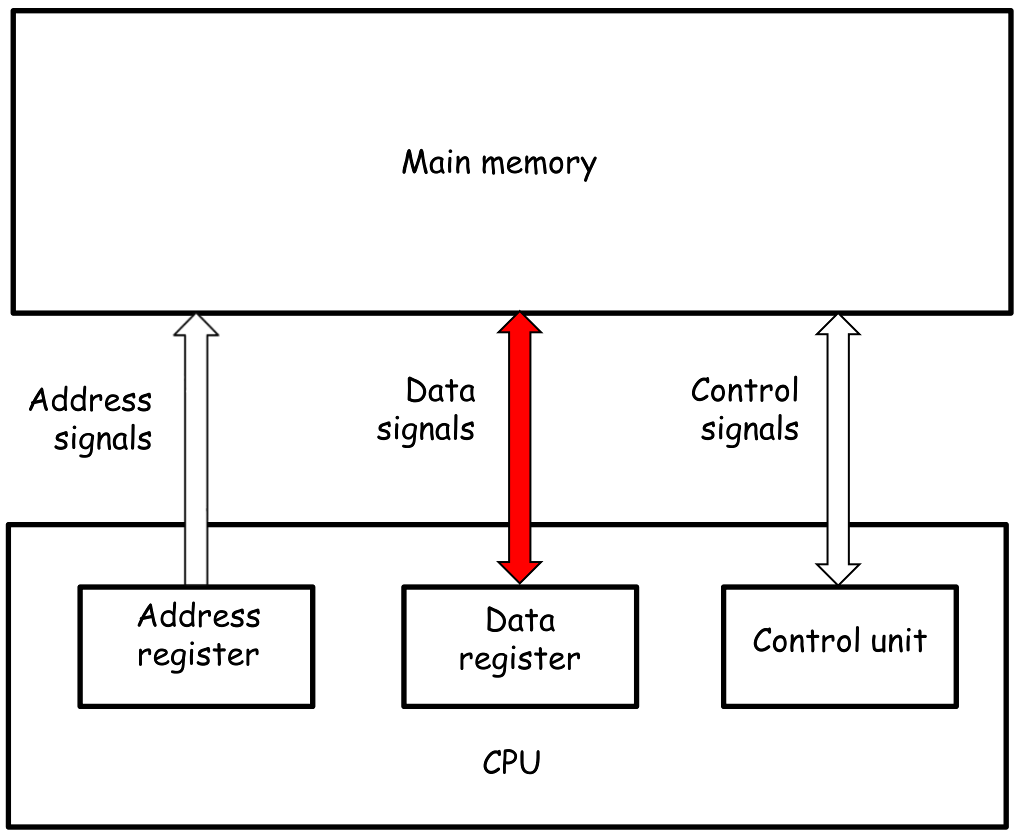
How does CPU access the main memory?

Example for accessing instructions:

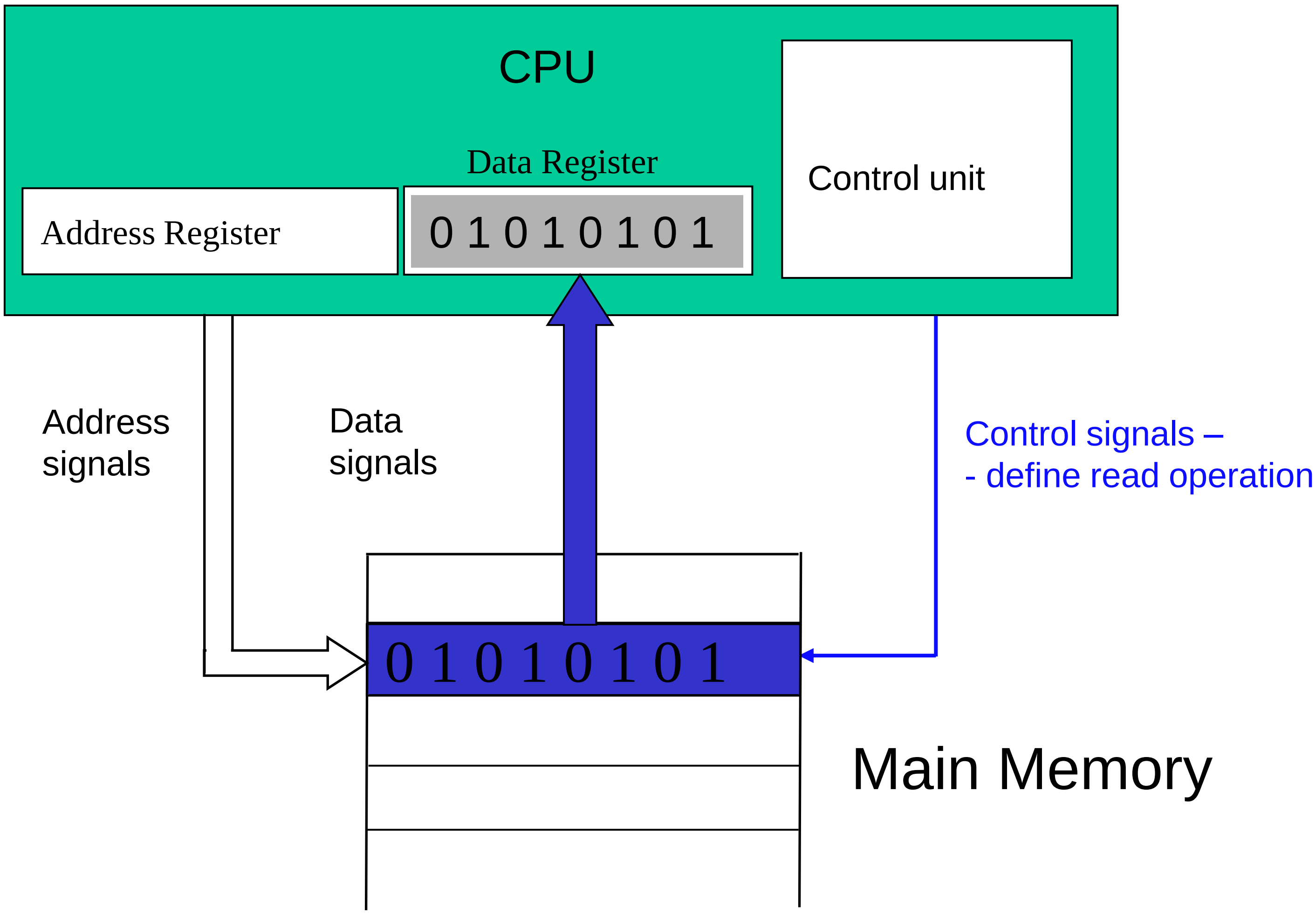


How does CPU access the main memory?

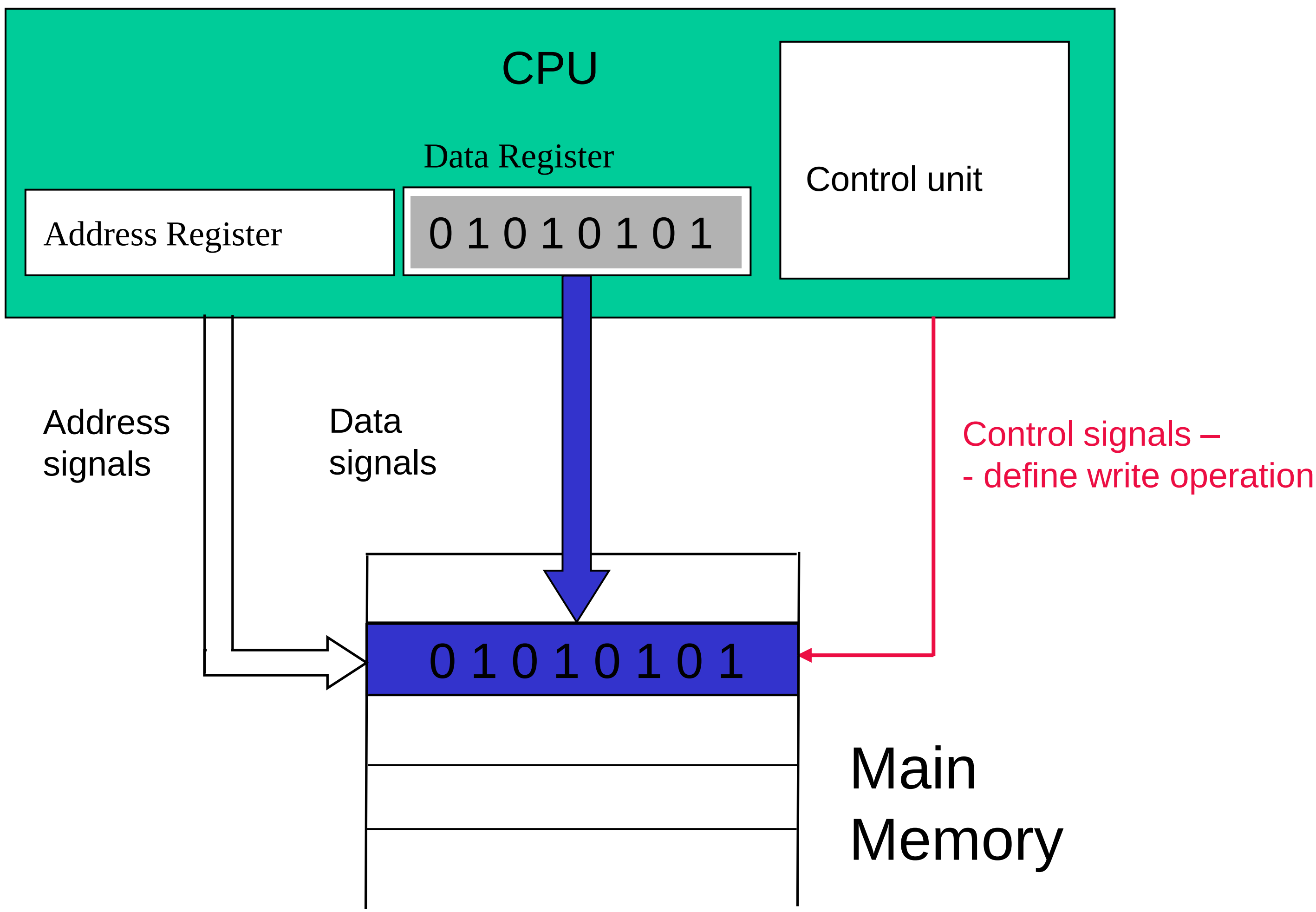
Examples for accessing operands:



CPU and Main Memory – read access



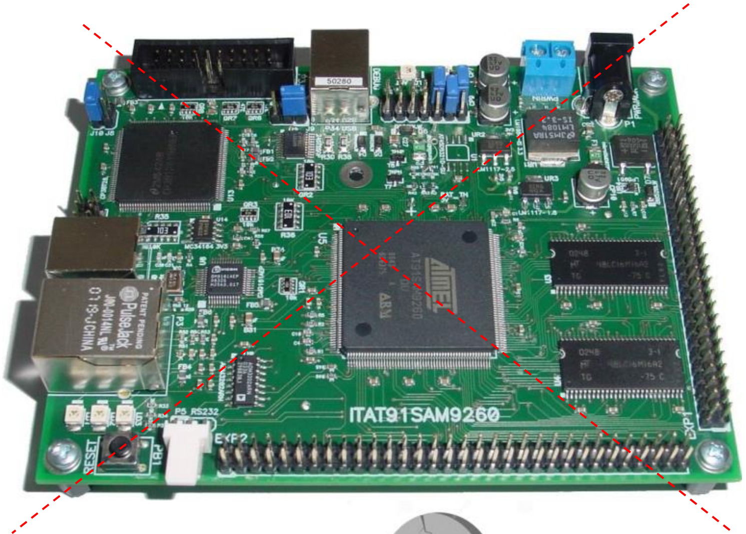
CPU and Main Memory – write access



Computer architecture CA

Computer STM32H750-DK

- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9



LAB 1.4 Quick intro to numeral systems

LAB 1.4 Quick intro to numeral systems

Numeral systems - short intro

petek, 16. oktober 2020 08:50

123 $\stackrel{?}{=}$ VALUE

REPRESENTATION

NUMERAL SYSTEM

DECIMAL: (BASE 10) 0-9

$$123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 = 100 + 20 + 3 = \underline{\underline{123}}$$

BINARY: (BASE 2) BIN 0,1

$$0b111 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 4 + 2 + 1 = \underline{7}$$

HEXADECIMAL: (BASE 16) HEX

$$0x40 = 4 \cdot 16^1 + 0 \cdot 16^0 = \underline{\underline{64}}$$

RELATED

DIGITS

HEX	VALUE	BIN
0	0	0000
⋮	⋮	
9	9	
A	10	
B	⋮	
⋮	⋮	
F	15	1111

1 DIGIT HEX ⇔ 4 DIGITS IN BIN

SHORTER

0x40

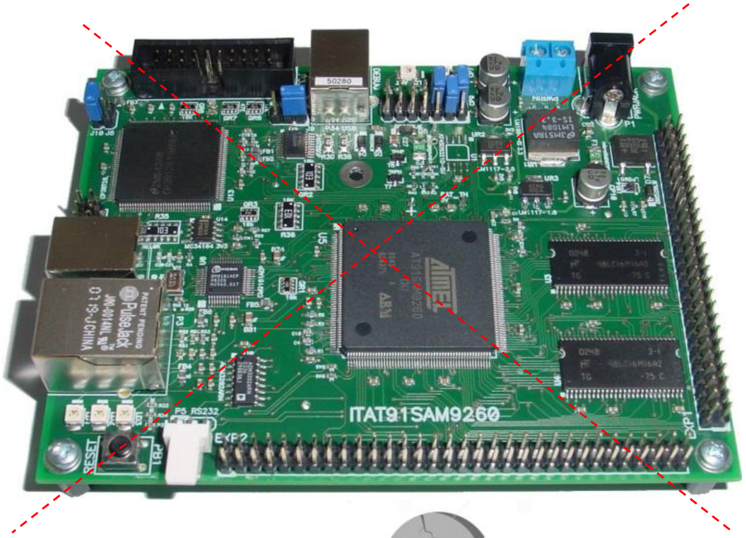
0b 0100 0000

8 4 2 1

Computer architecture CA

Computer STM32H750-DK

- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9

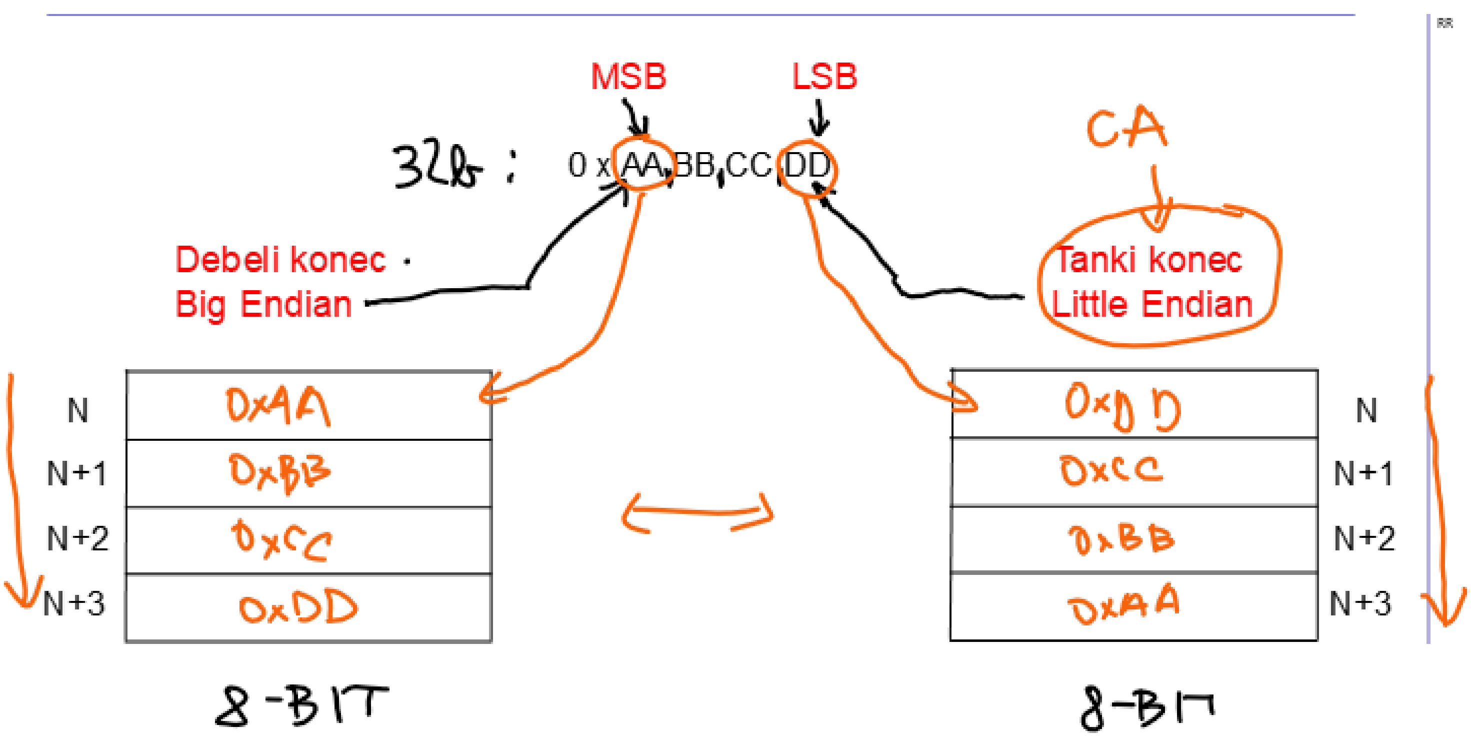


LAB 1.5 Big and Little Endian rules

LAB 1.5 Big and Little Endian rules

Big vs. Little Endian

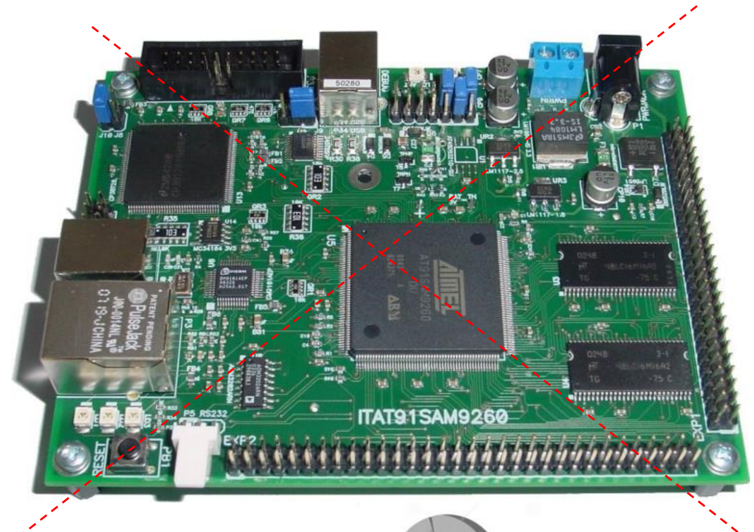
Monday, October 12, 2020 2:18 PM



Computer architecture CA

- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9

Computer STM32H750-DK



LAB 1.6 Addition – human, python, assembler cases

Human (case: 64 + 16 = 80)

$64 + 16 = ?$

$$\begin{array}{r} 64 \\ +\,{}_{1}16 \\ \hline 80 \end{array}$$

Python (case: REZ = STEV1 + STEV2)

Adding two variables in Python.

http://goo.gl/YXQ5qN

Python 2.7

```
STEV1=0x40
STEV2=0x10
REZ = STEV1 + STEV2
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. Use 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
adr r0, stev1	R0 ← Addr. of stev1	0xE24F0014
ldr r1, [r0]	R1 ← M[R0]	0xE5901000
adr r0, stev2	R0 ← Addr. of stev2	0xE24F0018
ldr r2, [r0]	R2 ← M[R0]	0xE5902000
add r3, r2, r1	R3 ← R1 + R2	0xE0823001
adr r0, rez	R0 ← Addr. of rez	0xE24F0020
str r3, [r0]	M[R0] ← R3	0xE5803000

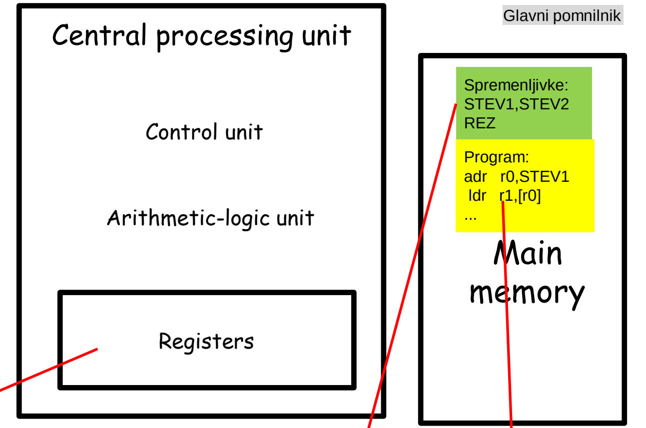
Execute instructions step-by-step and observe the register's values and the variable's values inside the main memory.

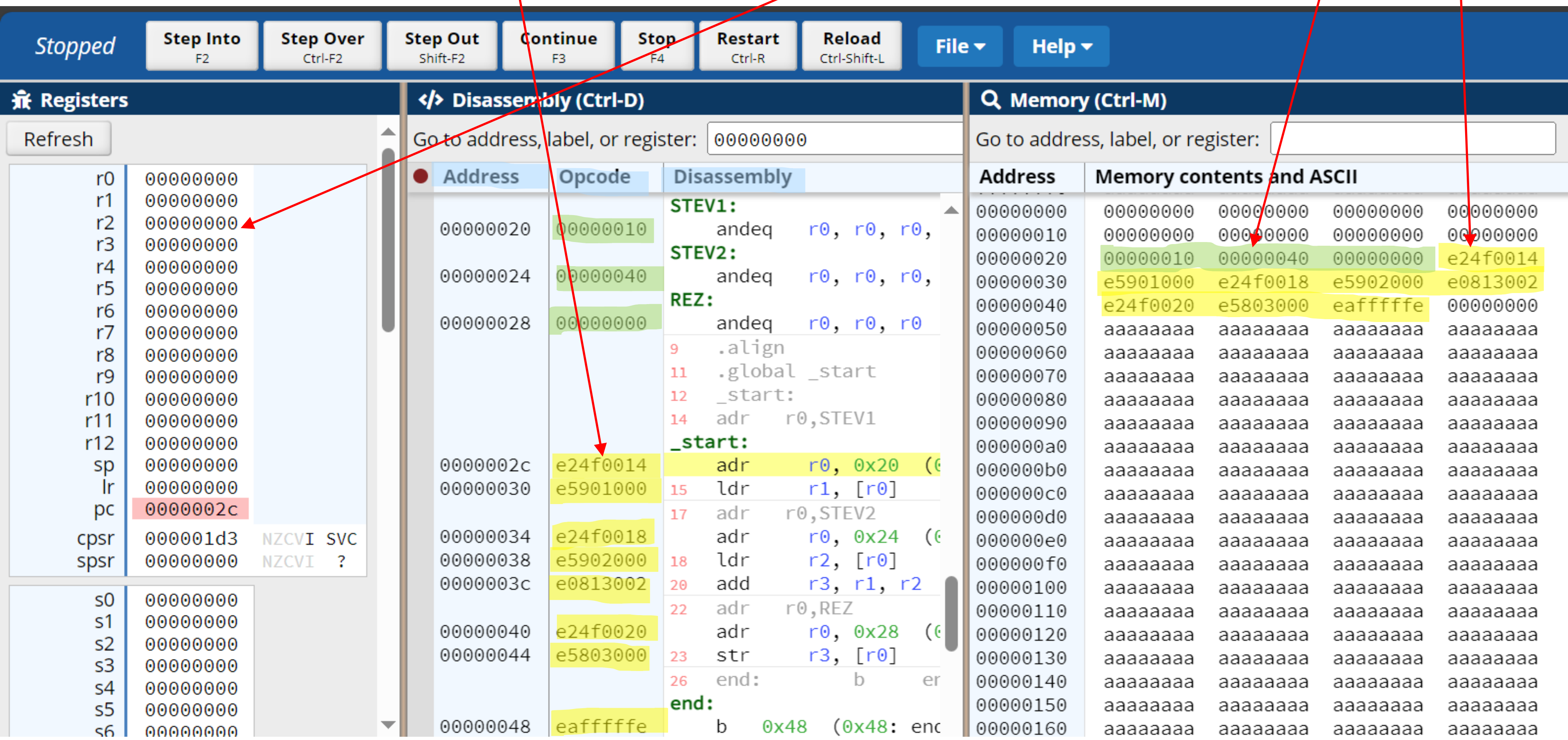
Practical example : Sum of 2 numbers

https://cpulator.01xz.net/?sys=arm&loadasm=share/s8zU3xx.s

Assembly language	Instruction description	Machine language
adr r0, stev1	R0 ← Addr. of stev1	0xE24F0014
ldr r1, [r0]	R1 ← M[R0]	0xE5901000
adr r0, stev2	R0 ← Addr. of stev2	0xE24F0018
ldr r2, [r0]	R2 ← M[R0]	0xE5902000
add r3, r2, r1	R3 ← R1 + R2	0xE0823001
adr r0, rez	R0 ← Addr. of rez	0xE24F0020
str r3, [r0]	M[R0] ← R3	0xE5803000

Assembler (compiler)

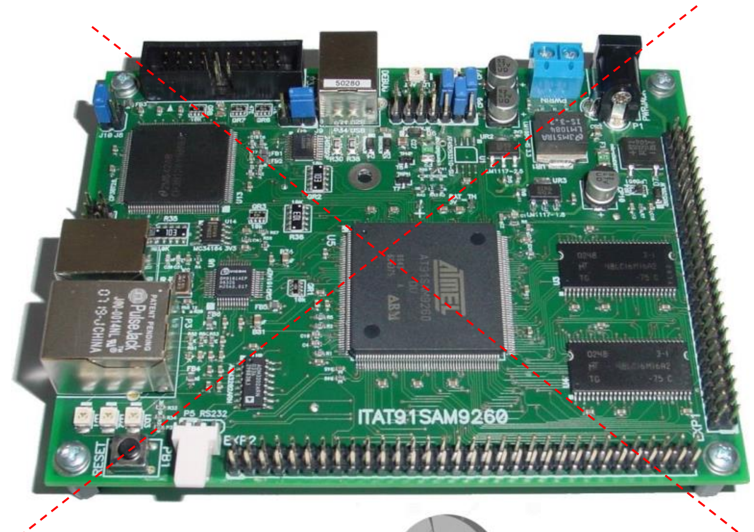




Computer architecture CA

Computer STM32H750-DK

- Računalnik FRI-SMS
  - Mikrokrmilnik AT91SAM9260 iz družine mikrokrmilnikov ARM9



LAB 1.7 Notes – empty templates

Python (case: REZ = STEV1 + STEV2)

Frames Objects

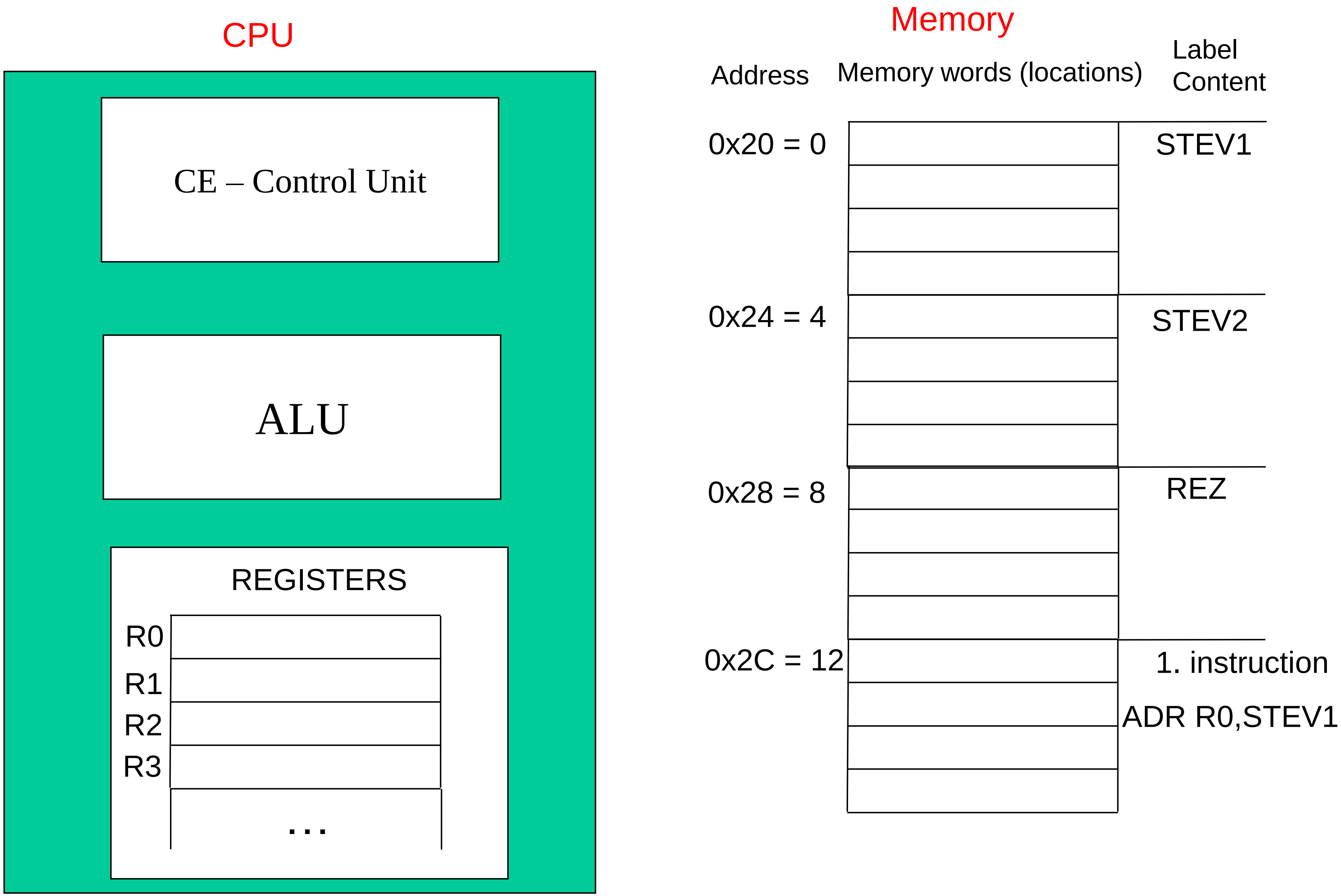
Global frame	
STEV1	64
STEV2	16
REZ	80

Python 2.7

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

http://goo.gl/YXQ5qN

Zgled: adding two numbers



Zgled: adding two numbers

INSTRUCTIONS

	Machine Instr.	Assembly Instr.	Description	Comment
1.	0xE24F0014	adr r0, stev1	R0 ← Addr. of stev1	
2.	0xE5901000	ldr r1, [r0]	R1 ← M[R0]	
3.	0xE24F0018	adr r0, stev2	R0 ← Addr. of stev2	
4.	0xE5902000	ldr r2, [r0]	R2 ← M[R0]	
5.	0xE0823001	add r3, r2, r1	R3 ← R1 + R2	
6.	0xE24F0020	adr r0, rez	R0 ← Addr. of rez	
7.	0xE5803000	str r3, [r0]	M[R0] ← 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