NATIONAL RESEARCH UNIVERSITY

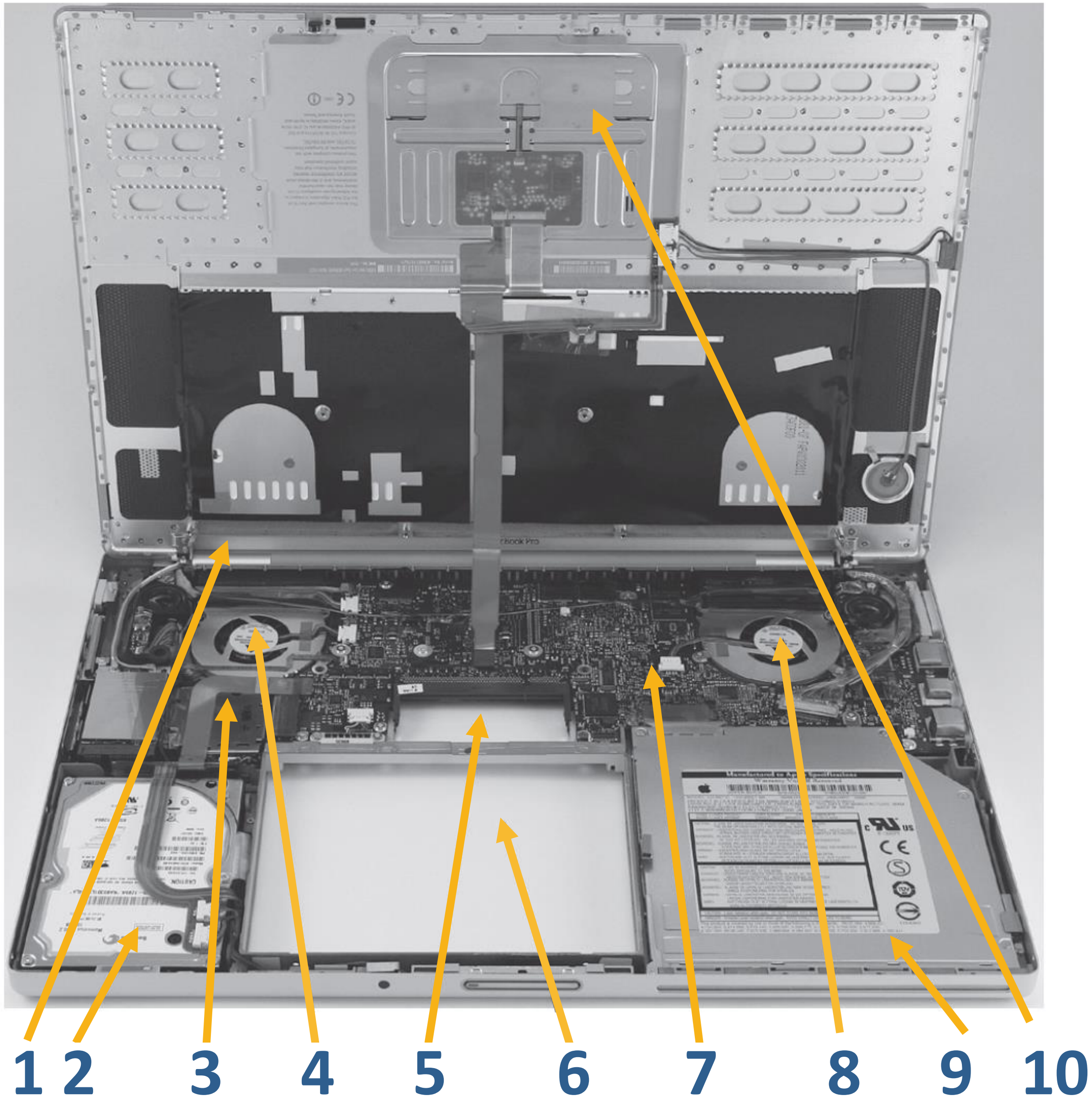
Faculty of Computer science
Higher School of Economics

Computer Architecture and Operating Systems
Lecture 3: Computer Architecture

Andrei Tatarnikov

atatarnikov@hse.ru
@andrewt0301

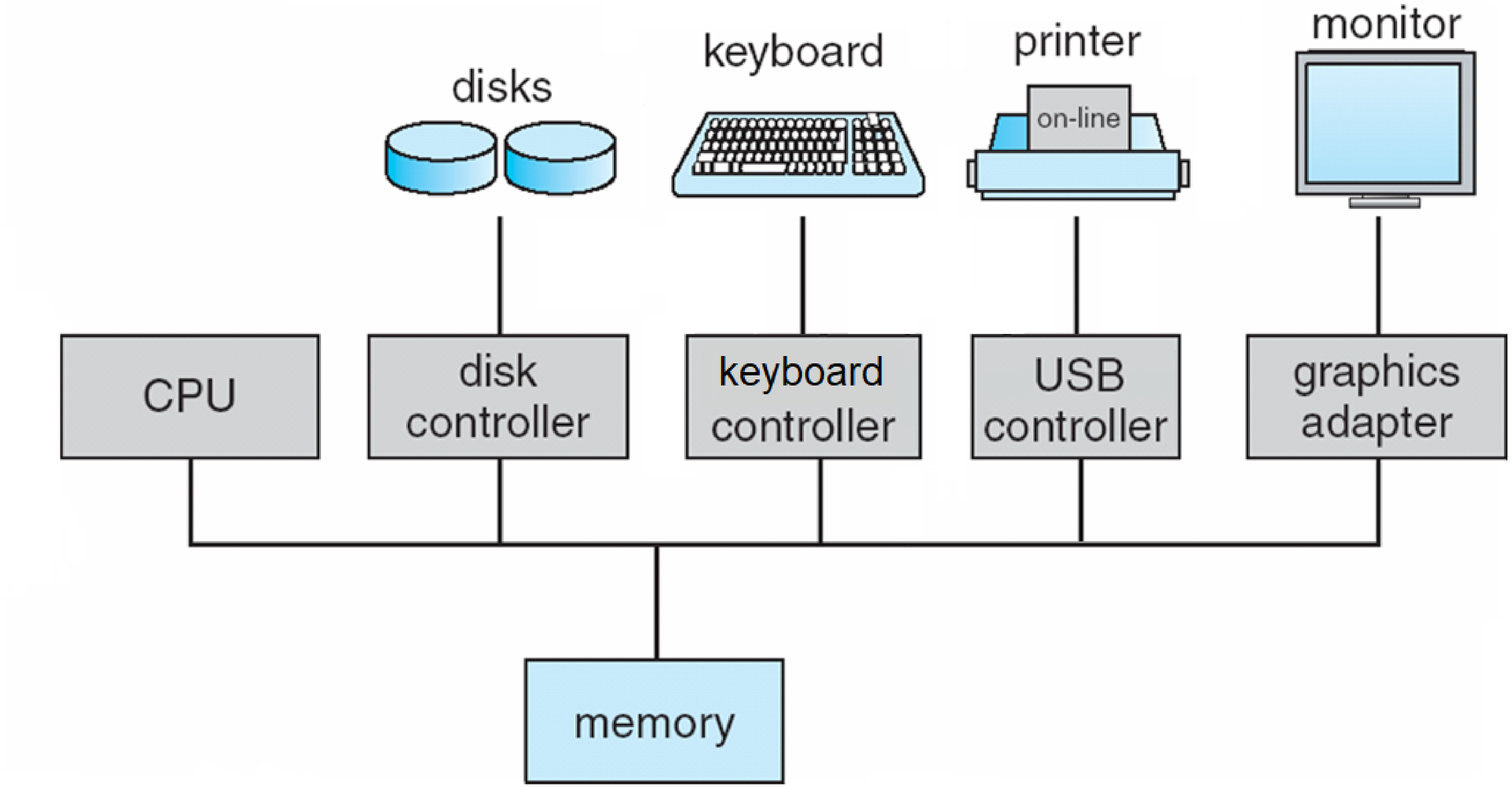
Computer Under Cover



1. Monitor
2. Hard drive
3. CPU (Processor)
4. Fan with cover
5. Spot for memory DIMMs
6. Spot for battery
7. Motherboard
8. Fan with cover
9. DVD drive
10. Keyboard

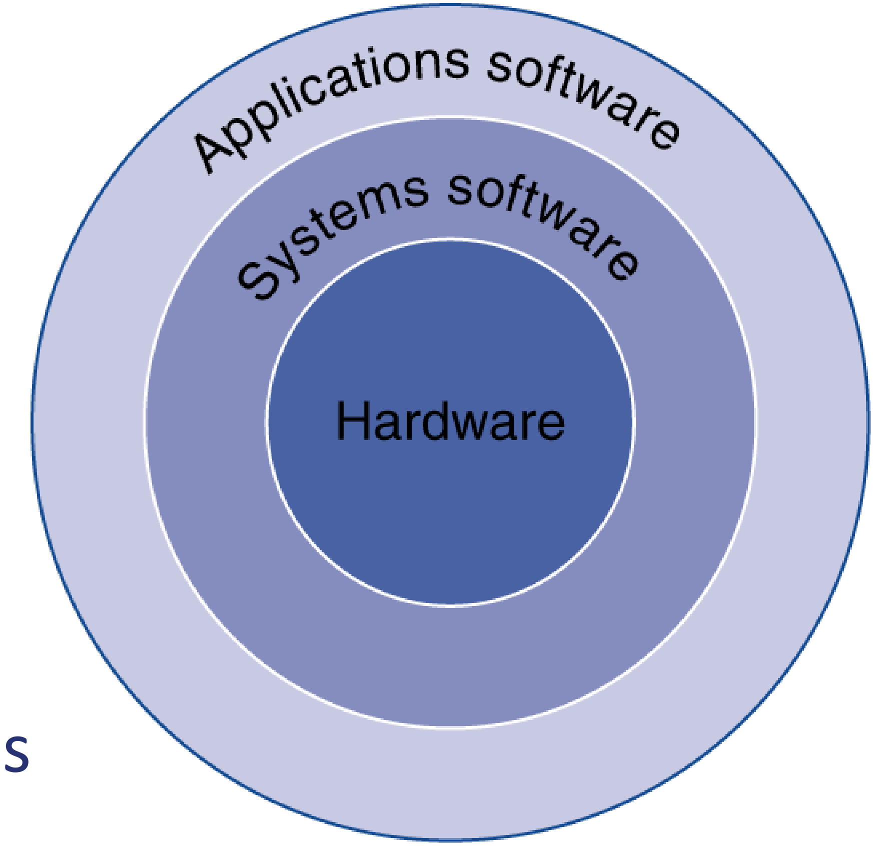
Computer Organization

- One or more CPUs and device controllers connected through a bus providing access to shared memory



Program Under Hood

- Application software
  - Written in high-level language
- System software
  - Compiler: translates high-level language code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - CPU, memory, I/O controllers



Levels of Program Code

High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler

Assembly language program (for RISC-V)

```
swap:
      slli x6, x11, 3
      add  x6, x10, x6
      ld   x5, 0(x6)
      ld   x7, 8(x6)
      sd   x7, 0(x6)
      sd   x5, 8(x6)
      jalr x0, 0(x1)
```

Assembler

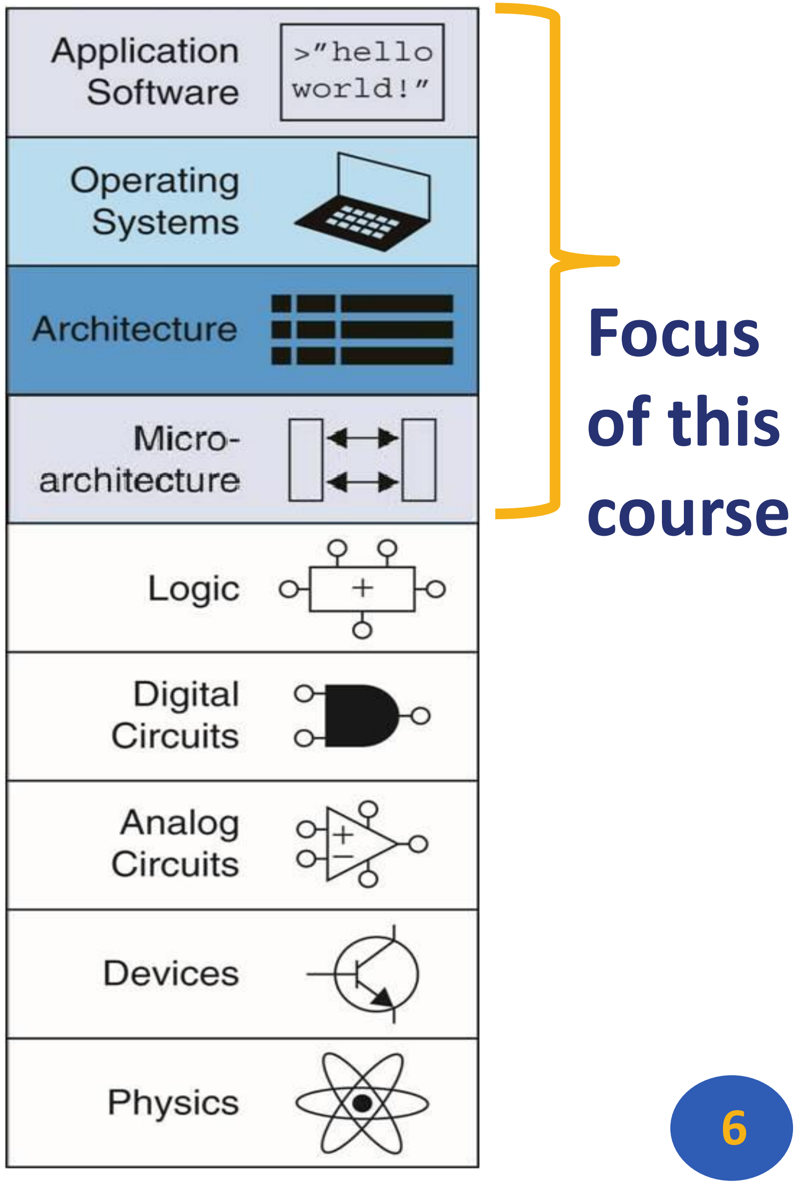
Binary machine language program (for RISC-V)

```
00000000001101011001001100010011
00000000011001010000001100110011
00000000000000110011001010000011
00000000100000110011001110000011
00000000011100110011000000100011
00000000010100110011010000100011
00000000000000001000000001100111
```

- High-level language
  - Level of abstraction closer to problem domain
  - Provides productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data

Abstractions

- Abstraction helps us deal with complexity
  - Hide lower-level detail
- Instruction set architecture (ISA)
  - The hardware/software interface
- Application binary interface (ABI)
  - The ISA plus system software interface
- Implementation (microarchitecture)
  - The details underlying the interface

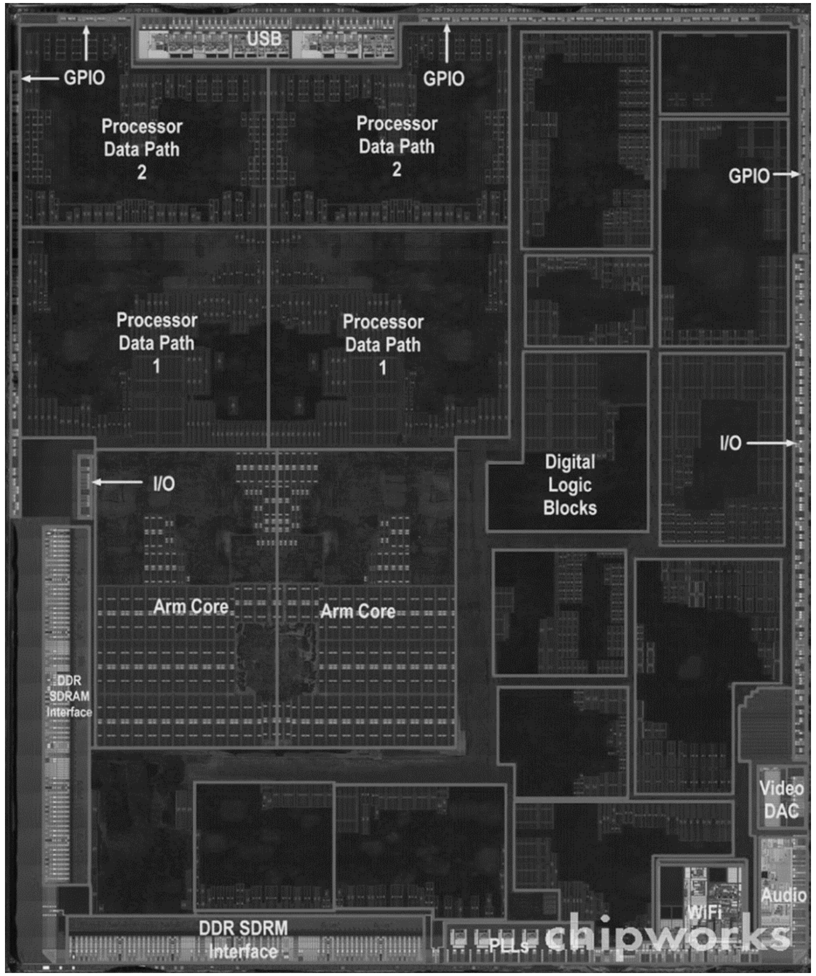


Inside the Processor (CPU)

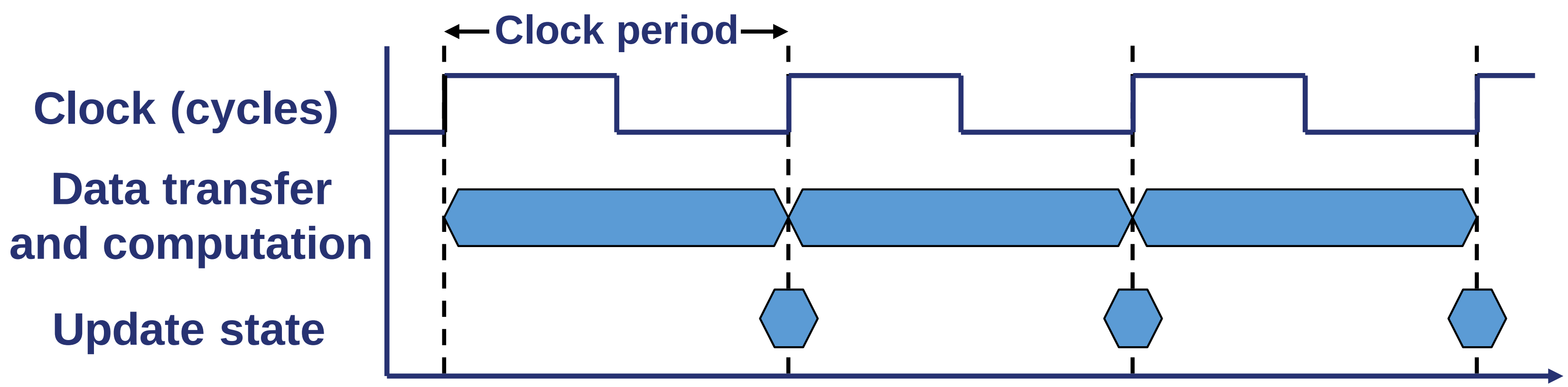
Central Processing Unit (CPU) is the heart of any computer system.

Main components:

- **Register file:** small fast memory for immediate access to data
- **Datapath:** performs operations on data
- **Control unit**: sequences datapath, memory, etc.



CPU Clocking



- Operation of digital hardware governed by a constant-rate clock
- Clock period: duration of a clock cycle
  - e.g., 250 ps = 0.25 ns = 250×10^{-12} s
- Clock frequency (rate): cycles per second
  - e.g., 4.0 GHz = 4000 MHz = 4.0×10^{9} Hz

CPU Time

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Instruction Set Architecture (ISA)

Instruction set architecture (ISA) is the interface between the hardware and the lowest-level software. This is one of the most important abstractions.

ISA Classification

- Complex instruction set computer (CISC)
  - x86/x64 (Intel and AMD)
- Reduced instruction set computer (RISC)
  - ARM, PowerPC, MIPS, RISC-V
- Very long instruction word (VLIW)
  - Itanium, Elbrus

Reduced Instruction Set Computing (RISC)

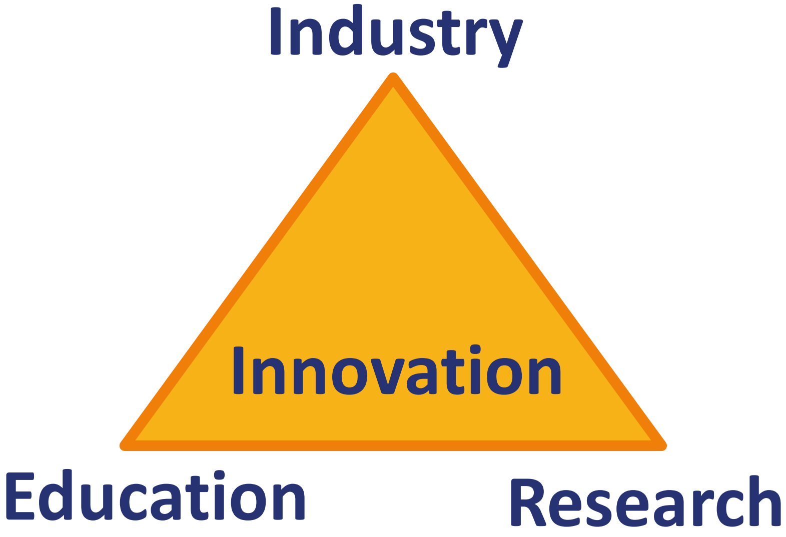
Reduced Instruction Set Computing (RISC) concept was proposed by teams of researchers at **Stanford University** (John Hennessy) and **University of California Berkeley** (David Paterson) in **early 1980s** as an alternative of Complex Instruction Set Computing (CISC) dominating at that time.

RISC Principles

- All instructions are executed by hardware
- Maximize the rate at which instructions are issued
- Instructions should be easy to decode
- Only loads and stores should reference memory
- Provide plenty of registers

RISC-V ISA

- Simple ISA by UC Berkeley (2010)
- Open and Free
- Wide-Purpose Configurable ISA (from IoT to mainframes)
- Maintained by RISC-V Foundation (moved to Switzerland)
- Supported by many IT Companies and Universities



RISC-V Community

Wide Support of IT Companies (except Intel and ARM) and Universities

Alibaba Cloud

HUAWEI

ANDES TECHNOLOGY

StarFive 赛昉科技

Western Digital.

NVIDIA.

SAMSUNG

NXP

ETH zürich

IBM

Google

and many others...

Instruction Execution

1. Fetch next instruction from memory into instruction register
2. Change program counter to point to next instruction
3. Determine type of instruction just fetched
4. If instructions uses word in memory, determine where Fetch word, if needed, into CPU register
5. Execute the instruction
6. Go to step 1 to begin executing following instruction

Any Questions?

```
                .text
    __start:    addi t1, zero, 0x18
                addi t2, zero, 0x21
    cycle:      beq t1, t2, done
                slt t0, t1, t2
                bne t0, zero, if_less
                nop
                sub t1, t1, t2
                j cycle
                nop
    if_less:    sub t2, t2, t1
                j cycle
    done:       add t3, t1, zero
```