1 ARM

Definition 1.1: ARM/LEG

Two types of data both stored in RAM:

- Instructions (4-byte blocks called **words**)
  - Instruction address is a multiple of four

    Notation. Address: instruction
- Data (8-byte blocks called **double words**)

An ARM program can access 2^{64} bytes of RAM, numbered from 0 to $2^{64} - 1$. When data is needed for calculations, it is loaded into **registers** that are far quicker to access.

Definition 1.2: Registers

There are 32 registers that can be used like a programming variable, but via ARM instruction.

- Each register stores 64 bits/8 bytes

  Notation. X0, X1, ..., X31
- X31 (XZR) always contains 0

32 registers is clearly not enough. There are special instructions to access RAM:

- **LDUR** - load unscaled register
- **STUR** - store unscaled register

  Remark. Unscaled means the address is exactly as specified (can access non-multiples of 4)

Proposition 1.3: Instructions

Hardware can only execute extremely simple instructions. There are five general types of ARM instructions that operate on registers or RAM. Format refers to how many and what type of operands.

- **R-format** (register)

  Example. ADD X1, X2, X3 adds the contents of X2 and X3 and stores the value in X1.
- **D-format** (data)

  Example. LDUR X1, [X2, #20] loads data from memory address X2+20 into register X1

  Example. STUR X1, [X2, #30] stores data from X1 into memory at address X2+30
- **I-format** (immediate)

  Example. ADDI X1, X1, #1 adds *immediate* value 1 to X1, stores result in X1 (X1++)
- **B-format** (branch)

  Example. B #28 sets program counter to PC+4*28
- **CB-format** (conditional branch)

  Example. CBZ X1, #8 if X1 == 0 then set program counter to PC+4*8

Definition 1.4: Program Counter (PC)

- No if, for, while, etc. statements in ARM. Instead, control flow is handled by goto commands:
  - branch (unconditional goto)
  - CBZ or CBNZ (conditional goto)
- Special register, *program counter* (PC) stores address of executing instruction
- Whenever a non-goto instruction is executed, PC increments by 4

2 Computers

Proposition 2.1: Computer components (DOMIC)

- Datapath: performs arithmetic operations
- Output: e.g. speaker
- Memory: Stores the running program
- Input: e.g. microphone
- Control: Tells the datapath what to do

Remark. CPU = Datapath + Control

Definition 2.2: ISA

The **instruction set architecture (architecture)** is the interface between the hardware and the lowest-level software. It includes everything needed to make a program work.
Example. A complete machine language, I/O, allocating memory, etc.
Remark. Has many *implementations* varying in cost and performance for the same software.

Definition 2.3: ABI

The **application binary interface** is a basic set of machine instructions + the OS interface to programmers. The OS takes care of I/O, allocating memory, and other low-level system functions.

Definition 2.4: Volatile

Memory that requires power to maintain its stored information.

Main/primary memory is volatile, holds data and programs in memory while they run.
Example. DRAM
Secondary memory is **non-volatile**, stores information between runs
Example. Flash memory, magnetic/hard disks

Definition 2.5: Transistor

A digital on/off switch. An **integrated circuit (IC)**, made from silicon, combines lots of transistors.
Remark. Moore's Law states the number of transistors in a dense IC doubles about every two years.

Definition 2.6: Performance

There are two important measures of performance:

Lemma 2.7: Response time

Also called latency or execution time. The time between the start and completion of a task.

Lemma 2.8: Throughput

Also called bandwidth. The total amount of work done in a given time.

Remark. Improving response time usually improves throughput.

Throughput is typically more important, but sometimes we want to prioritize response time (web servers: each individual's experience as fast as possible over maximum simultaneous people served)

Proposition 2.9: Factors affecting program performance

- Algorithms
  Example. Number of source-level statements, I/O operations
- Language, compilers, architecture
  Example. Number of computer instructions for each source-level statement
- Processor and memory
  Example. How quickly instructions are executed
- IO system
  Example. How quickly I/O is executed

Definition 2.10: Clock

All computers have a clock:

- Clock ticks: high voltage point in frequency wave
- Clock cycles: time between consecutive ticks
- Rate $= \frac{1}{\text{cycle}}$ (i.e. ticks per second)

Proposition 2.11: Factors affecting response time

- Clock speed (GHz)
- Complexity of instruction set (how many lines it takes to perform a task)
- Efficiency of compilers (redundant code deleted, more than one operation in a cycle)
- Mix of instructions needed to complete a task (float vs int operations)
- Some choices in designing computers (IPC):
  - Simple instruction set, fast clock, one instruction executed per clock cycle
  - Complex instruction set, slower clock, one instruction executed per cycle

Remark. Simple instructions take as long as complicated ones

- Complex instruction set, faster clock, some instructions take multiple cycles to execute
- Maximize average instructions per cycle (IPC)

Definition 2.12: Benchmarks

We could measure performance with:

- Real programs
- Synthetic benchmarks
- Micro-benchmarks

Synthetic benchmarks are most generally applicable:

- Standard set of programs and data chosen to measure performance
  Example. Standard Performance Evaluation Corporation (SPEC)
- Advantages
  - Provides basis for meaningful comparisons
  - Design by committee may eliminate vendor bias
- Disadvantages
  - Vendors can optimize for benchmark performance
  - Possible mismatch between benchmark and user needs
  - Still an artificial measurement

Definition 2.13: Logic Blocks

- Combinationial: without memory (same output given same input)
- Sequential: with memory (possibly different outputs depending on memory)
  Example. Output 1 if input is eventually stable, possible with storage
- Inputs and outputs are 0 or 1