

Hack Memory

([Usage hints](#)  for this presentation)

IT Systems, Summer Term 2026
Dr.-Ing. Matthes Elstermann

(Author and License Information)

Speaker notes

Our next ambitious goal lies in the definition of a machine language for the Hack platform. Before we approach that goal, we take a brief look at memory in Hack. In this course, we do not build memory chips ourselves, but we take them for granted and only look at their functionality.

1. Introduction

Speaker notes

Let us begin with a brief look at the core of our topic and its learning objectives.

1.1. Today's Core Question

- What are bits, bytes, words, registers, RAM, ROM?
 - Based on Chapter 3 of ([Nisan and Schocken 2005](#))
 - (We skip Project 3)

Speaker notes

This presentation introduces basic terminology related to main memory in computers, in particular bits, bytes, words, registers, RAM, and ROM.

However, Project 3 is not part of our course. Thus, we do not build memory chips but take them for granted.

1.2. Retrieval Practice

- Recall the [Hack computer architecture preview](#) 
 - What types of memory exist in Hack for what purposes?
 - How do memory, ALU, and PC interact?

Speaker notes

Please take a brief break and recall how the Hack computer architecture looks like, with its types of memory, the ALU as active component, and the program counter as pointer for the next instruction to execute.

1.3. Learning Objectives

- Explain memory hierarchy
- Explain functionality of RAM, ROM, PC
 - In particular for Hack chips
- Convert memory sizes (bits, bytes, larger sizes)

Speaker notes

Please think about the learning objectives for this presentation. Which ones did you achieve already, maybe based on skills outside this course?

Missing ones are covered subsequently.

Agenda

- 1. Introduction
- 2. Memory
- 3. Hack Memory
- 4. Conclusions

Speaker notes

The short agenda of this presentation is as follows. After this introduction, we look at computer memory in general, before we turn to memory in our Hack computer.

2. Memory

Speaker notes

In memory, our computers store data and instructions.

2.1. Sequential Circuits

- Previously, **combinational circuits** 
 - Output depends on input combinations
 - Pure functions, e.g., Nand, ALU
- Now, **sequential** circuits: Output depends on input combination, **clock signal**, and **state**
 - Calculations based on **sequence** of (previous) inputs
 - Sequence triggered/synchronized by clock: one step or **cycle** per clock tick
- Memory chips, RAM, ROM
- Computer with RAM and CPU

Previously, we designed combinational circuits that compute functions of their input arguments.

Now, with memory, we consider sequential circuits where the output depends not only on the current input arguments but also on a clock signal and state, where the state itself may depend on a sequence of previous inputs.

The clock of a processor generates pulses which synchronize the inner workings of a computer, for modern CPUs typically measured at the rate, or frequency, of gigahertz. With a rate of one gigahertz, a single CPU step takes one nanosecond, for a billion steps per second. The steps are also called cycles.

We now look at the functionality of typical memory chips, which in combination with the CPU form the major parts of computers.

2.2. Bit, Byte, Word

- **Bit** (b): Smallest unit, zero or one
- **Word**: CPU specific width for unit of processing
 - Hack: 16 bits, more typical: 32 or 64 bits
- **Byte** (B): 8 bits, **typical** unit of **addressing** in modern CPUs
 - But **not** in Hack architecture
 - In Hack, addresses refer to 16-bit words, not to 8-bit bytes
 - Byte can be understood as unsigned 8-bit integer
 - Between 0 and 255

Speaker notes

This slide lists terminology regarding bits, bytes, and words. We already saw the bit as basic binary unit. Several bits are grouped into a word, whose size depends on the computer architecture, with 16 bits forming a word in the Hack architecture and 32 or 64 bits as typical sizes for modern processors.

Universally, a byte consists of 8 bits. Importantly, and as source of frequent confusion, modern memory is addressed by bytes, while Hack memory is addressed by words. Thus, in everyday memory, address 0 refers to the first byte, i.e., to the first 8 bits, while in Hack it refers to the first word, i.e., the first 16 bits. Address 1 then refers to the second byte in everyday memory, but to the second word in Hack. And so on.

2.2.1. Units for Memory

- Base 10 **and** base 2 for larger (or smaller) quantities
 - Systeme international d'unités (SI)
 - Prefixes based on **powers of 10**
 - “Normal”: kilo/k (10^3), mega/M (), micro/ μ (), nano/n (), ...
 - 16 kB = 16,000 B (16 kilobytes)
 - IEC (International Electrotechnical Commission)
 - Prefixes based on **powers of 2**
 - “New”: kibi/Ki (), mebi/Mi (), gibi/Gi (), ...
 - KiB = 2¹⁰ B (16 kibibytes)
- Typical convention
 - For RAM (main memory), use powers of 2
 - For disks (secondary memory), use powers of 10

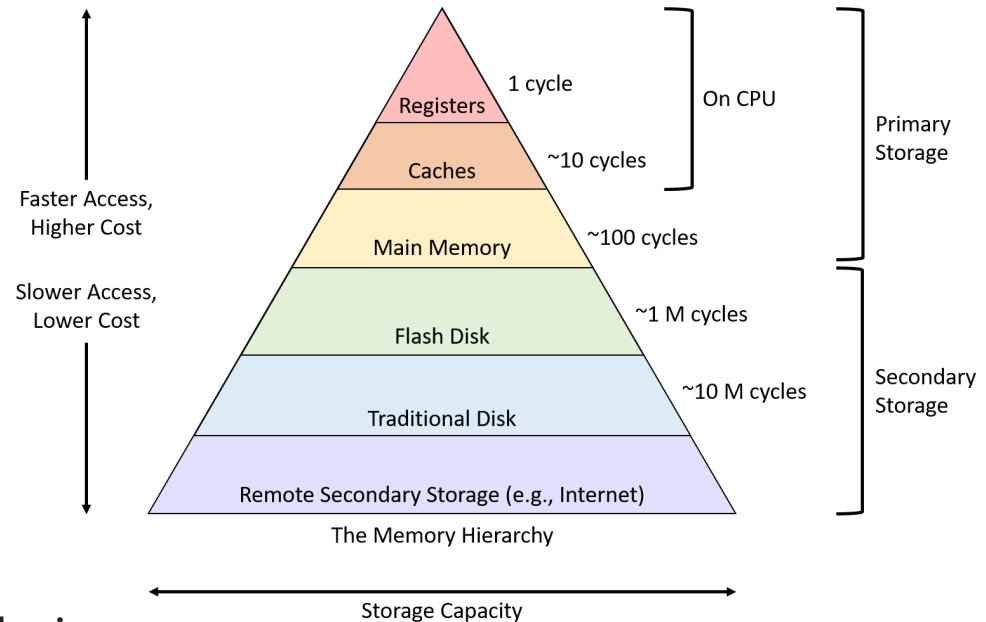
Speaker notes

You know typical units for large quantities from everyday life, e.g., a kilogram for 1000 grams. However, a kilobyte may mean different things in different contexts, either 1000 or 1024 bytes. To specify unambiguously what is meant, different prefixes exist that distinguish units based on powers of 10 from those based on powers of 2. Thus, a kibibyte is the proper way to talk about 1024 bytes.

Note that typically powers of 2 are used in the context of main memory (where bit strings of a fixed size are used to address bytes in main memory, giving rise to powers of 2 naturally), while powers of 10 are typical for secondary memory.

2.3. Memory Hierarchy

- **Registers:** Small memory, in CPU
 - Each register stores a word
 - Small, expensive, fast (< 1 ns access)
- **RAM** (main/primary memory): larger memory, on mainboard
 - Think of array of registers
 - **Address** input determines what register to access
 - Byte-addressed in real RAM, word-addressed in Hack
 - Read/write accesses possible in **random** order, at equal speed
 - Still fast (100 ns access time)
- CPU **caches:** Hierarchy of memory devices
 - Copy of recently used data between RAM and CPU
 - Level 1, L1: closest, smallest, fastest (1 ns access)
 - Level 2, L2: farther away, larger, slower (4 ns access)
- **Secondary** memory: Flash/solid-state disks (16 µs), traditional/hard disks (several ms)
 - Much larger and slower, **persistent**, cheap



"The memory hierarchy" Copyright (C) 2020 Dive into Systems, LLC under CC BY-NC-ND 4.0; from Dive into Systems

(Source for numbers [↗](#))

Memory of modern computers can be arranged in the hierarchy shown here, with registers being the smallest, fastest, and most expensive pieces of memory that are directly embedded into processors. Each register stores just one word but can be accessed within a single CPU cycle as it is located directly inside the CPU.

RAM is considerably larger, e.g., on the order of gigabytes, but also about a factor of 100 slower. Thus, if a machine instruction requires access to data in main memory, that instruction will be about a factor of 100 slower than a similar instruction with data in registers. In other words, it might require 100 CPU cycles in contrast to 1 cycle. The reason for this slowdown is that data needs to be fetched through a so-called [bus](#)  from RAM, covering a relatively large distance.

To amortize the cost of transferring data from RAM, recently accessed data is stored in so-called cache memory, which is located directly on the CPU chip. Subsequent accesses to data in caches is much faster than access to RAM. Typically, several levels of caches of increasing size and delay are located between CPU and RAM.

In addition to the memory types discussed so far, our computers usually also contain secondary memory on different types of disks or even on remote machines. Such memory is persistent in the sense that it keeps its contents even when powered off.

Note that in this hierarchy each level can be used as cache for memory below it: If data is copied from a lower level into a higher level, subsequent accesses benefit from the increased speed of the higher level. As memory at higher levels is relatively small, one needs to select carefully what to store. We will revisit this thought in the context of virtual memory for operating systems.

3. Hack Memory

Speaker notes

After these general thoughts on memory, let us now turn to specifics of the Hack computer.

3.1. Memory Chips

- Sequence of chips of increasing sizes

- Built-in DFF and Bit

- Single bit

- 16-bit Register for one word

- RAM8, RAM64, RAM512, RAM4K, RAM16K

- RAM chips of increasing sizes (8 words, 64 words, ...)
 - With inputs `in`, `address`, `load`; output `out`
 - If `load` is 1, store `in` at `address`; otherwise, keep state
 - Output current contents at `address` as `out`
- RAM16K: words

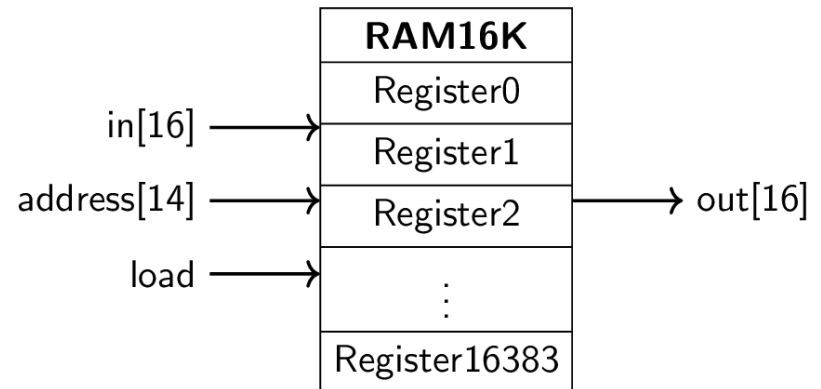


Figure under CC0 1.0

Speaker notes

Memory for the Hack computer can be built in a sequence of chips in Project 3 of Nand2Tetris, starting from a single bit over registers to larger chips. We skip that project.

What you need to understand, though, is the following:

RAM chips can be understood as arrays of registers, where an `address` input selects the particular register whose contents should be read or written. The number of `address` bits determines the size of the chip, e.g., 14-bit addresses for a chip with 16384 registers.

Each memory chip has a `load` bit, which specifies whether the current input should be stored as new contents for the addressed register or not.

3.2. Hack ROM

- Nand2Tetris with built-in chip ROM32K
 - Input `address[15]`
 - Output `out[16]`
 - (words)
- Output current contents at `address` as `out`
 - Only reading, contents written upon fabrication
 - (Later: ROM initialized with instructions when program is loaded)

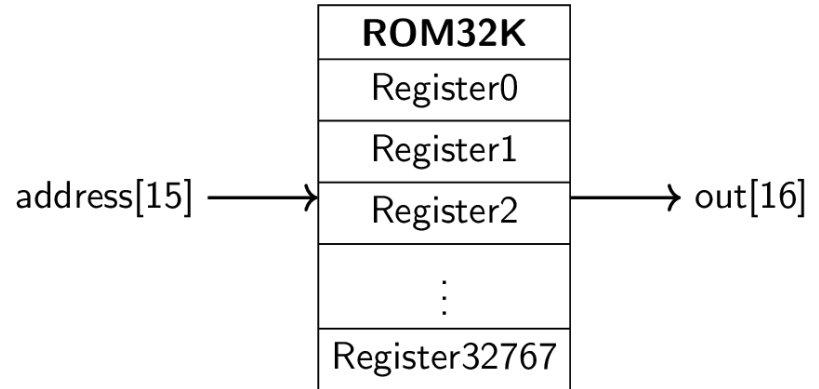


Figure under CC0 1.0

Speaker notes

Nand2Tetris comes with a builtin ROM chip, whose details you can see [here](#). Differently from RAM, it only has an `address` input as its contents cannot be changed programmatically.

3.3. Hack PC

- **Recall**: Program Counter (PC) holds address of next machine instruction to execute

- Inputs

- `in[16]`,
- Control bits: `reset`, `load`, `inc`

- Output `out[16]`

- Output current register contents as `out`

- Logic

```
if reset then store 0
elif load then store in
elif inc then increment stored value
else keep state
```

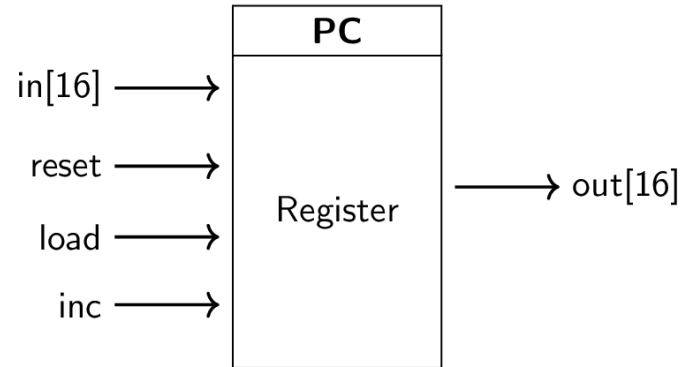


Figure under CC0 1.0

Speaker notes

The program counter, which will be embedded into the Hack CPU, is shown here. Essentially, it is a register that stores a counter value with functionality based on 3 control bits: It is possible to **reset** the counter to 0, to **load** a new value, to increment the current counter, or to keep the current state.

4. Conclusions

Speaker notes

Let us conclude.

4.1. Summary

- Memory organized in hierarchy of levels
 - Considerable differences regarding size, speed, cost
 - From registers over caches to RAM and secondary storage
 - Registers located inside CPU, including the program counter
- RAM can be understood as array of registers
 - RAM is byte-addressed in real hardware, word-addressed in Hack
- Outlook 
 - CPU includes registers
 - Computer includes registers, RAM, ROM

Speaker notes

Memory organization is structured in a hierarchy of levels, each exhibiting considerable disparities in size, speed, and cost. This hierarchy ranges from registers, including the program counter, built into the CPU, to caches, RAM, and secondary storage. RAM can be understood as array of registers. It is addressed by bytes in real hardware, but by words in the Hack architecture.

You will see how to build the CPU from ALU, registers, and program counter as well as the entire computer from CPU and memory in a [later presentation](#) .

4.2. Self-Study

- Play with RAM16K and PC in Hardware Simulator
 - (Use built-in chips)
 - Note
 - Area for RAM contents
 - For RAM16K, `address` enumerates 16-bit words, neither bits nor bytes
- How large is RAM16K in bits, bytes, words, KiB?

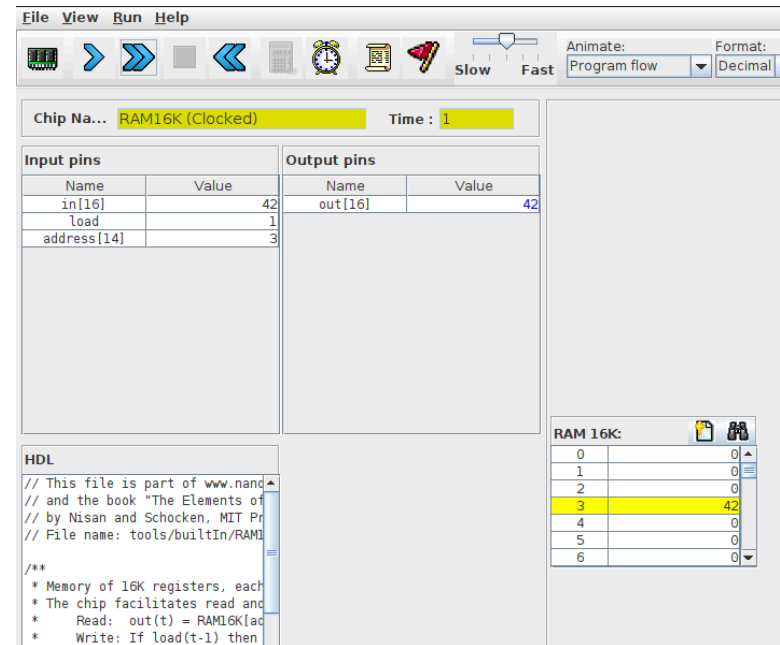


Figure under GPLv2

Speaker notes

Please take a break and experiment with the builtin chips for RAM and program counter in the Hardware Simulator.

Also recall the vocabulary regarding memory sizes.

Bibliography

Nisan, Noam, and Shimon Schocken. 2005. *The Elements of Computing Systems: Building a Modern Computer from First Principles*. The MIT Press. <https://www.nand2tetris.org/>.

Speaker notes

The bibliography contains references used in this presentation.

License Information

Source files are available on GitLab (check out embedded submodules) [↗](#) under free licenses [↗](#). Icons of custom controls are by [@fontawesome](#) [↗](#), released under CC BY 4.0 [↗](#).

Except where otherwise noted, the work “Hack Memory”, © 2024-2026 Jens Lechtenbörger [↗](#), is published under the Creative Commons license CC BY-SA 4.0 [↗](#).

Speaker notes

This presentation is distributed as Open Educational Resource under freedom granting license terms.

Source files are available on GitLab, where the author would be happy about contributions, e.g., in terms of issues and merge requests.

