DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

COMPUTER ARCHITECTURES

Information processing models

Prepared by: **Gábor Horváth**, ghorvath@hit.bme.hu
Presented by: **Gábor Lencse**, lencse@sze.hu

Győr,
09/14/2026

MŰEGYETEM 1782

ADMINISTRATIVE INFORMATION

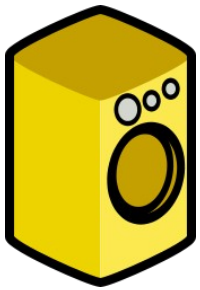
- Homepage: https://www.tilb.sze.hu/cgi-bin/tilb.cgi?0=m&1=targyak&2=GKNM_TATA112

- Requirements:
  - Two mid-term tests
    - Students must pass at least one of them to get the signature.
    - If both are at least 4, then recommended grade
  - Exam
    - For those who have the signature but do not have (or do not accept) recommended grade

WHY DO WE NEED THIS CLASS?

- We are going to understand
  - ...how a modern computer works
  - ...how a modern processor works
- „Professionals”
  - do not consider the computer to be a black box
  - understand how computers work
    → they can write **more efficient software**
  - know the compromises

WHAT IS A COMPUTER?



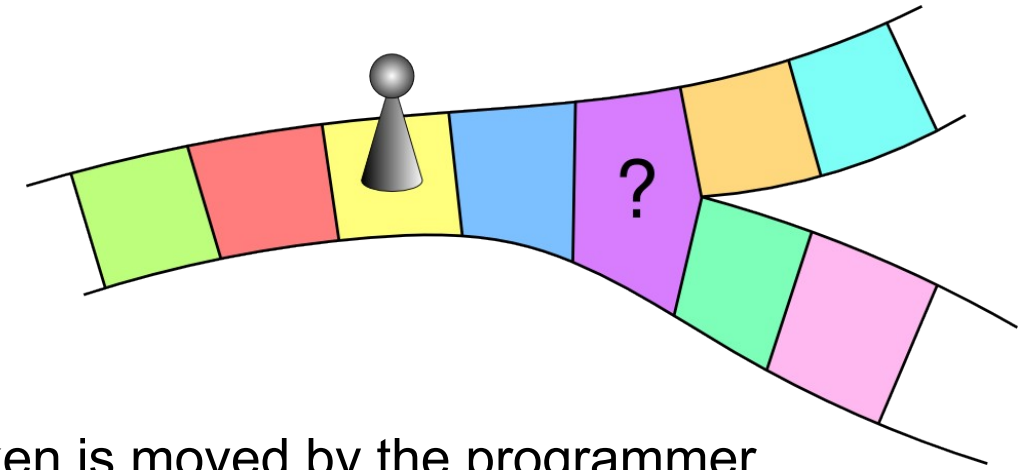
Information Processing Models

INFORMATION PROCESSING MODELS

- How to describe problems for a computer?
  - Control flow
  - Data flow
  - Demand driven

CONTROL-FLOW BASED MODEL

- Order of instructions: given by the control token

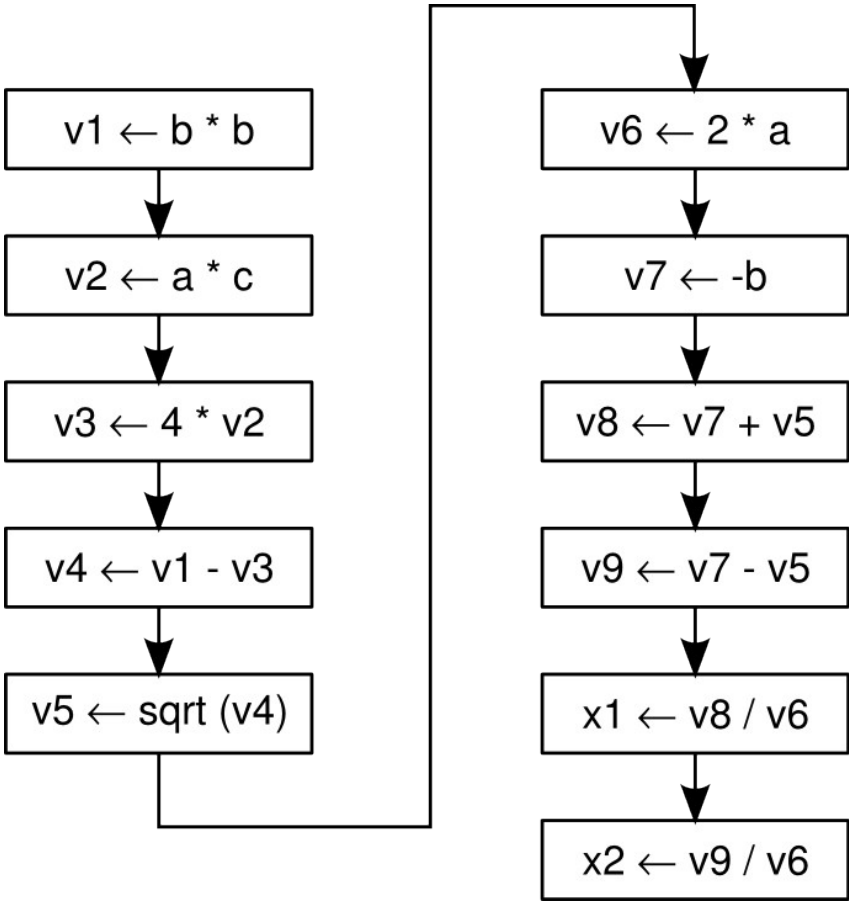


- Control token is moved by the programmer
  - Implicitly (to next instruction)
  - Explicitly (branch, goto, function call, return, etc.)
- Operands and results are stored in a shared memory
- Description of the program: *flow-chart*

CONTROL-FLOW BASED MODEL

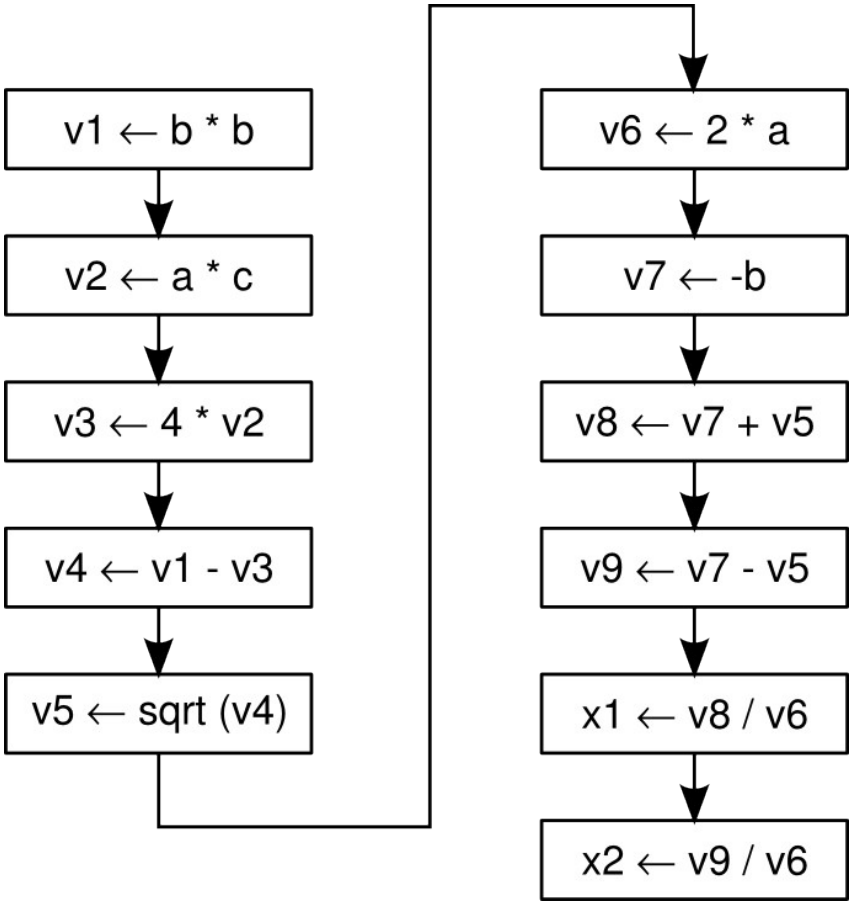
- Example: solutions of quadratic equations

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$



CONTROL-FLOW BASED MODEL

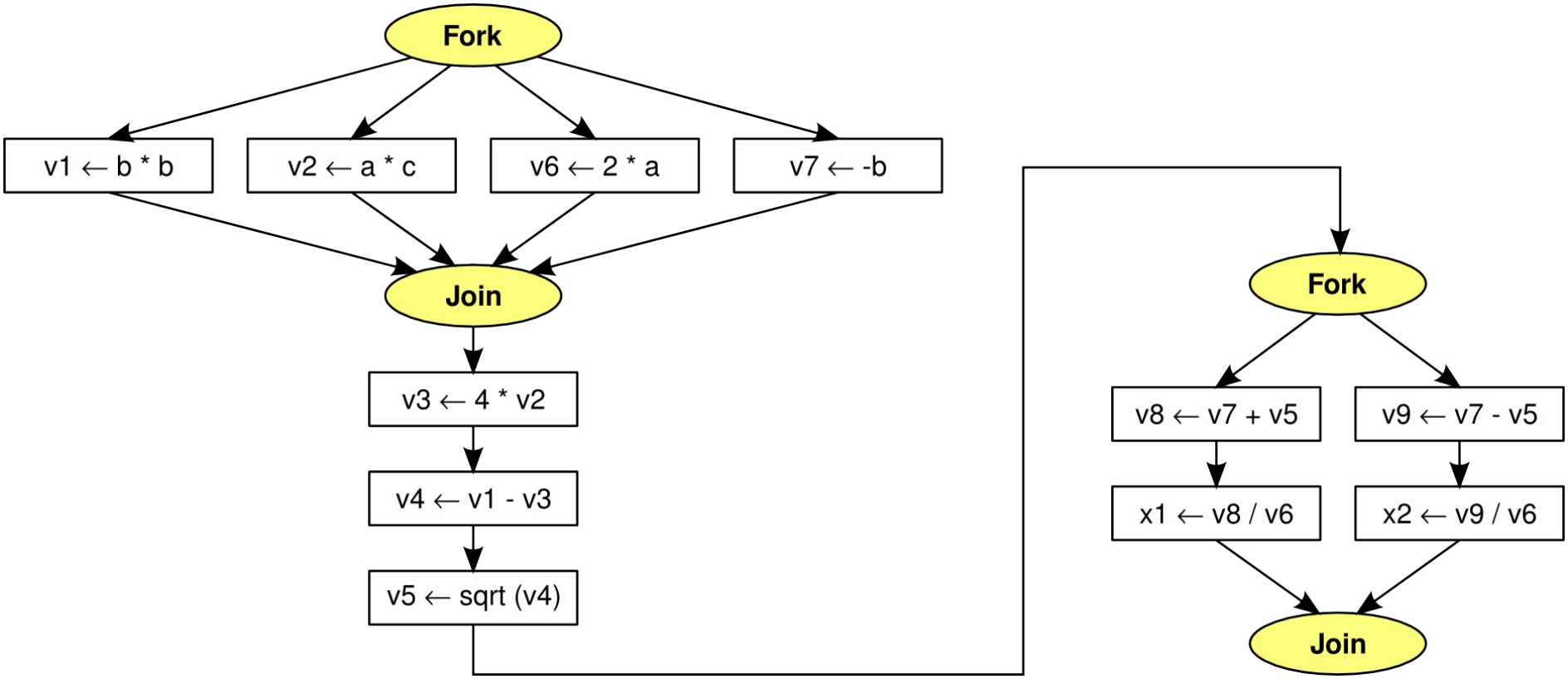
- Parallelism? Where?
  - v1, v2, v6, v7
  - v8, v9
  - x1, x2



CONTROL-FLOW BASED MODEL

- Parallelism can be described by
- Fork/Join primitives only
- **Fork**: creates multiple control tokens
- **Join**: joins control tokens

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

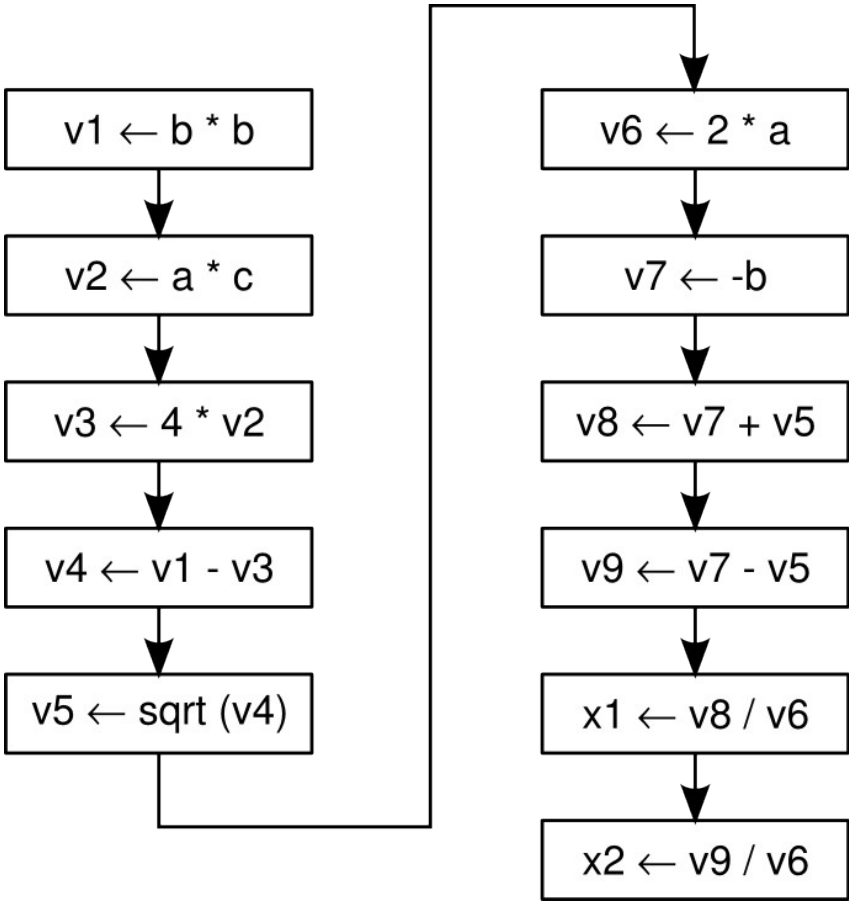


CONTROL-FLOW BASED MODEL

- Features
  - It can not detect parallelism
  - It can not exploit parallelism
  - Very wide-spread
  - Example: Neumann architecture
    - Control token: program counter
    - Shared memory: registers + system memory

DATA-FLOW MODEL

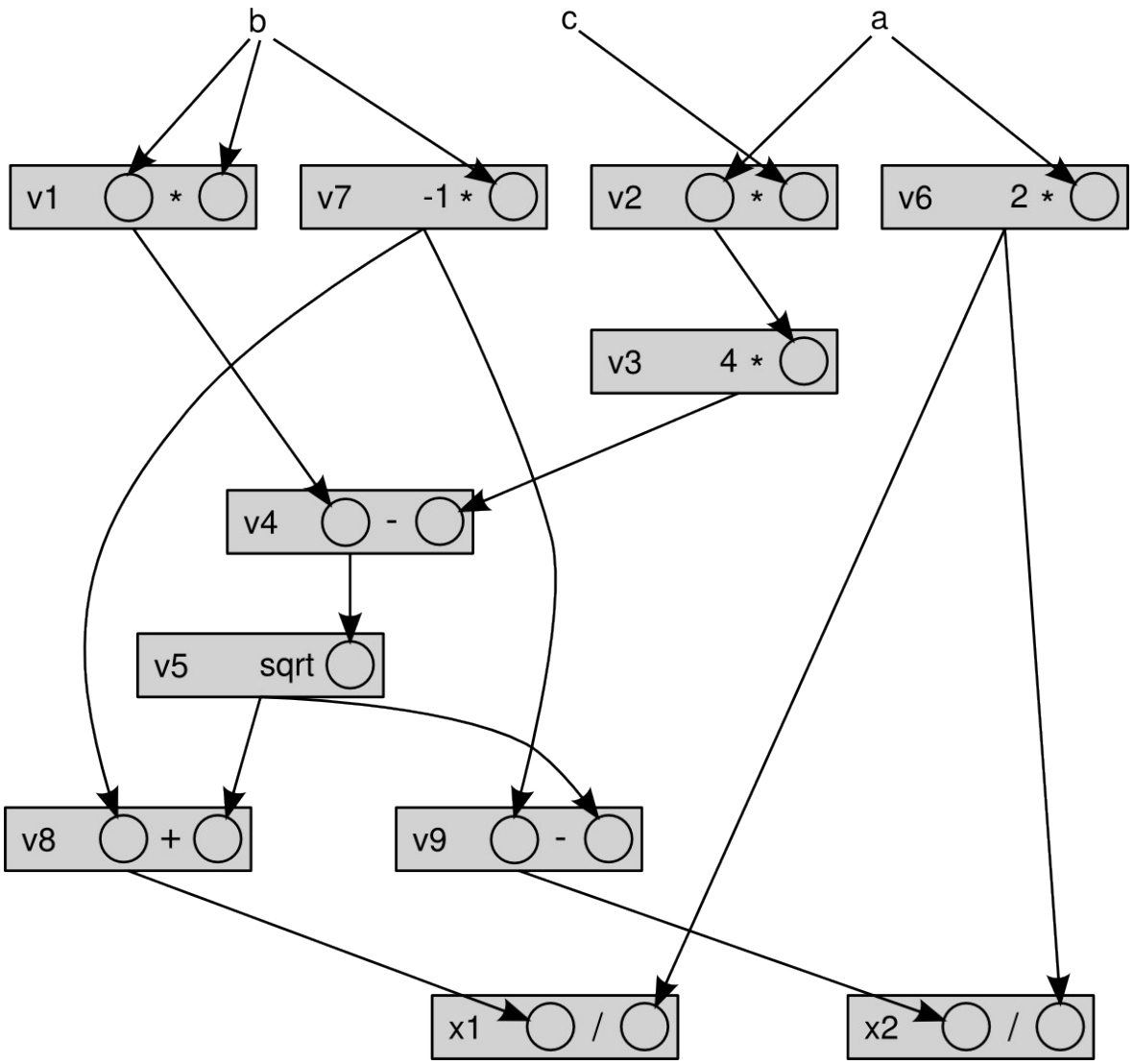
- How do we know that these can be executed in parallel?
  - v1, v2, v6, v7
  - v8, v9
  - x1, x2
- Dependence analysis!



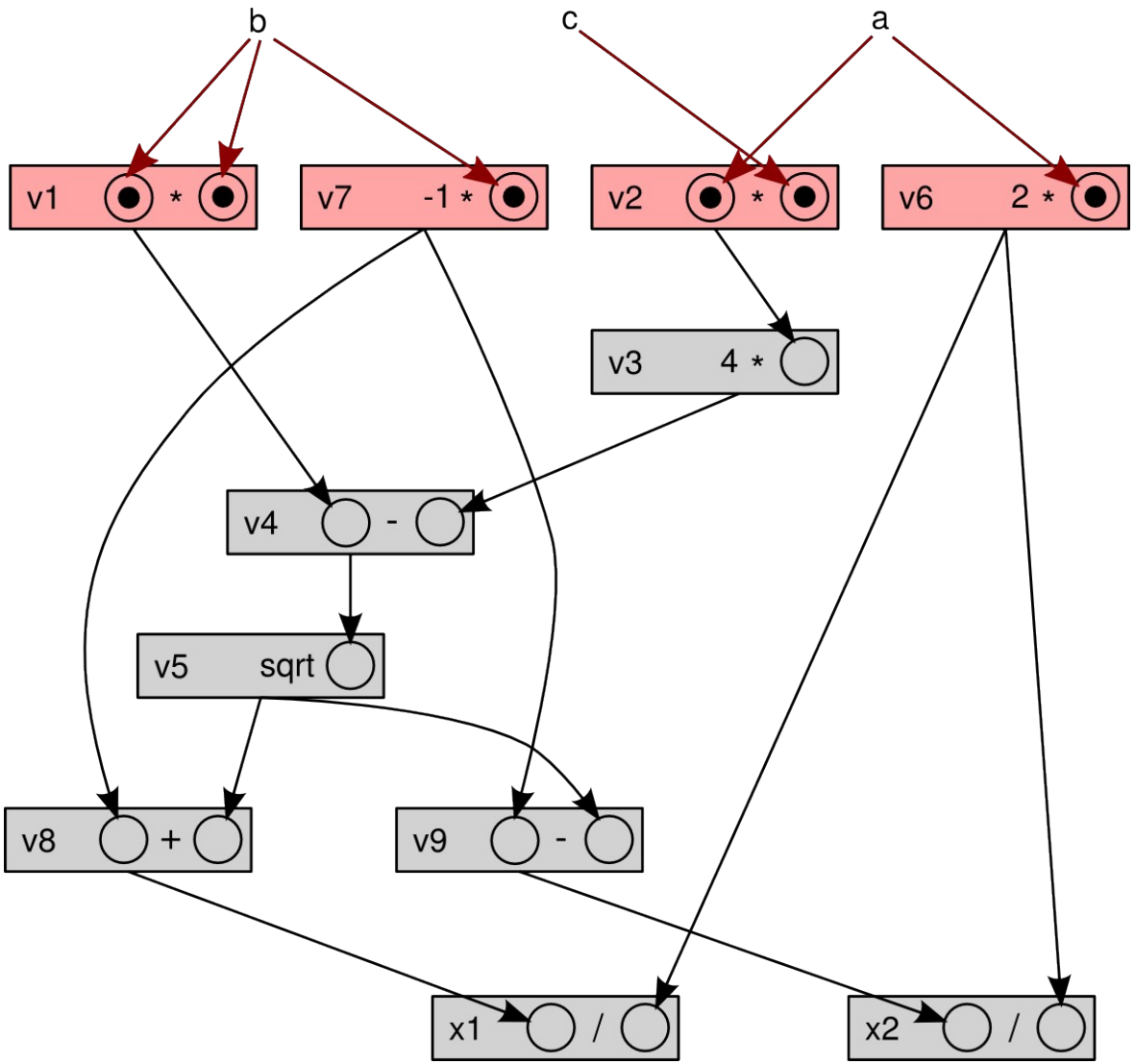
DATA-FLOW MODEL

- No control token
- No shared memory
- Instead of control token:
  - Execution of the instructions is automatically given:
    - **Every instruction is executed as soon as all its operands are available**
- Instead of shared memory:
  - Sub-results and operands of the instructions are passed directly among them
- Description of the program: *precedence graph*
- Example: solutions of quadratic equations

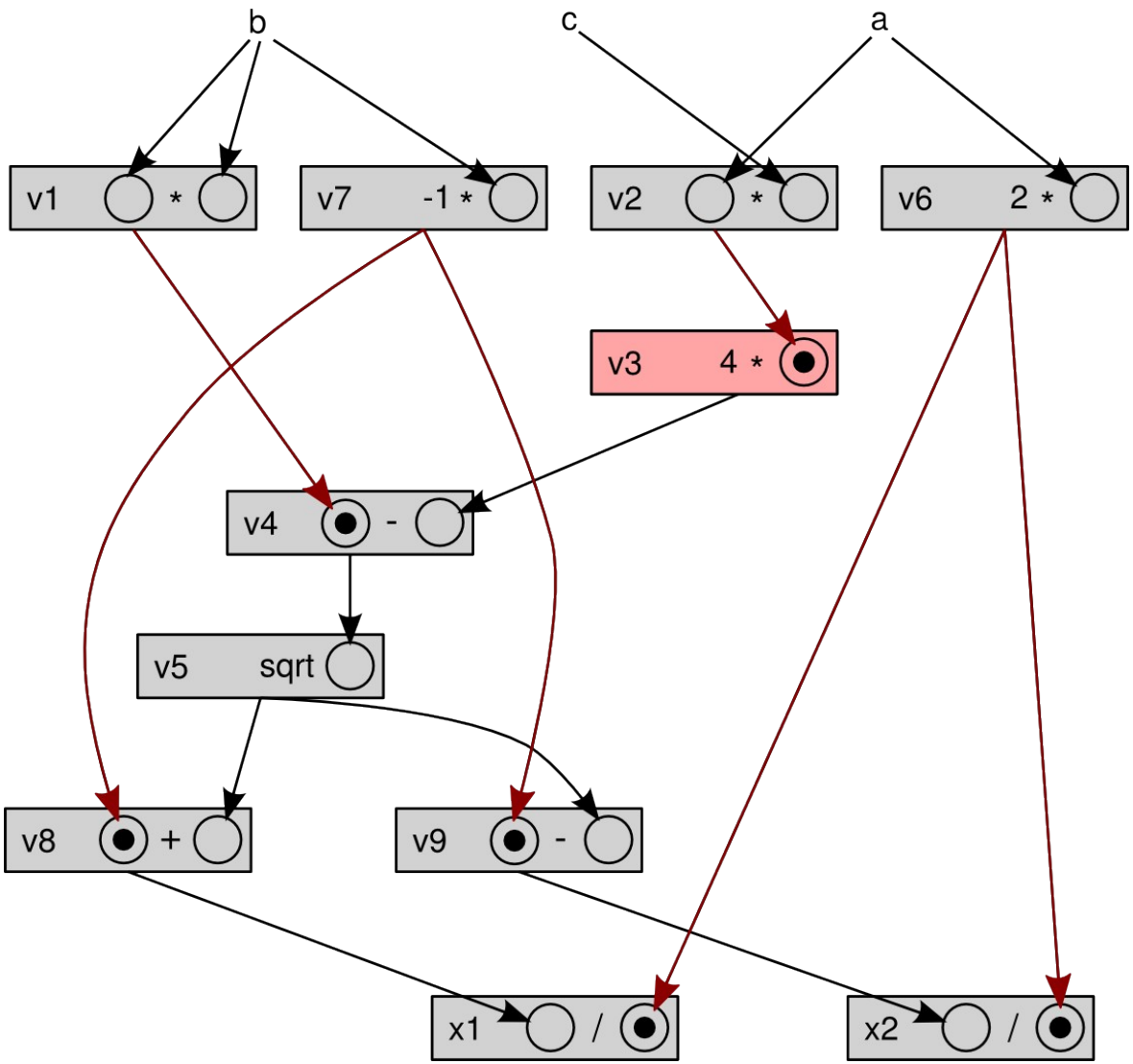
DATA-FLOW MODEL



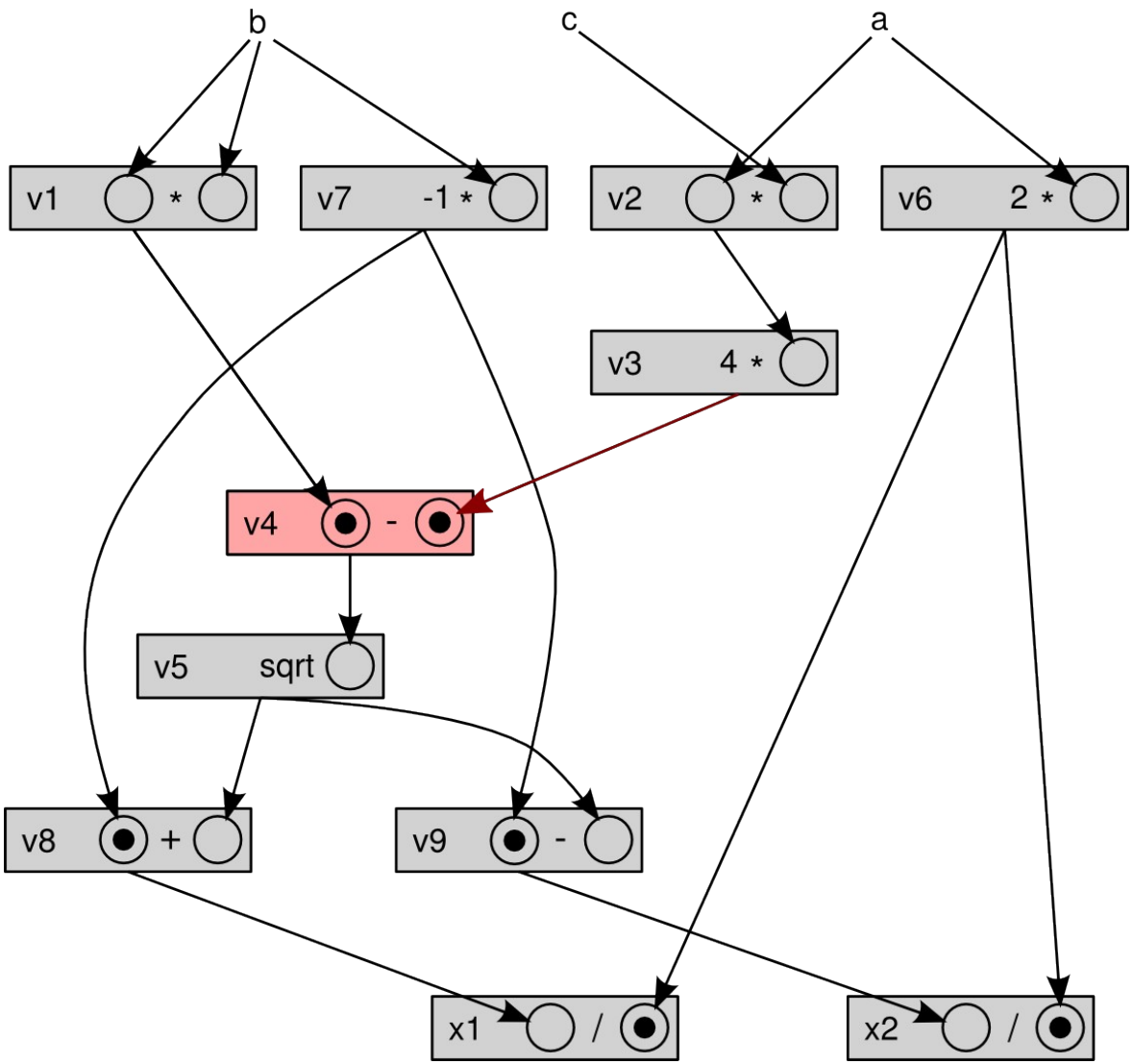
DATA-FLOW MODEL



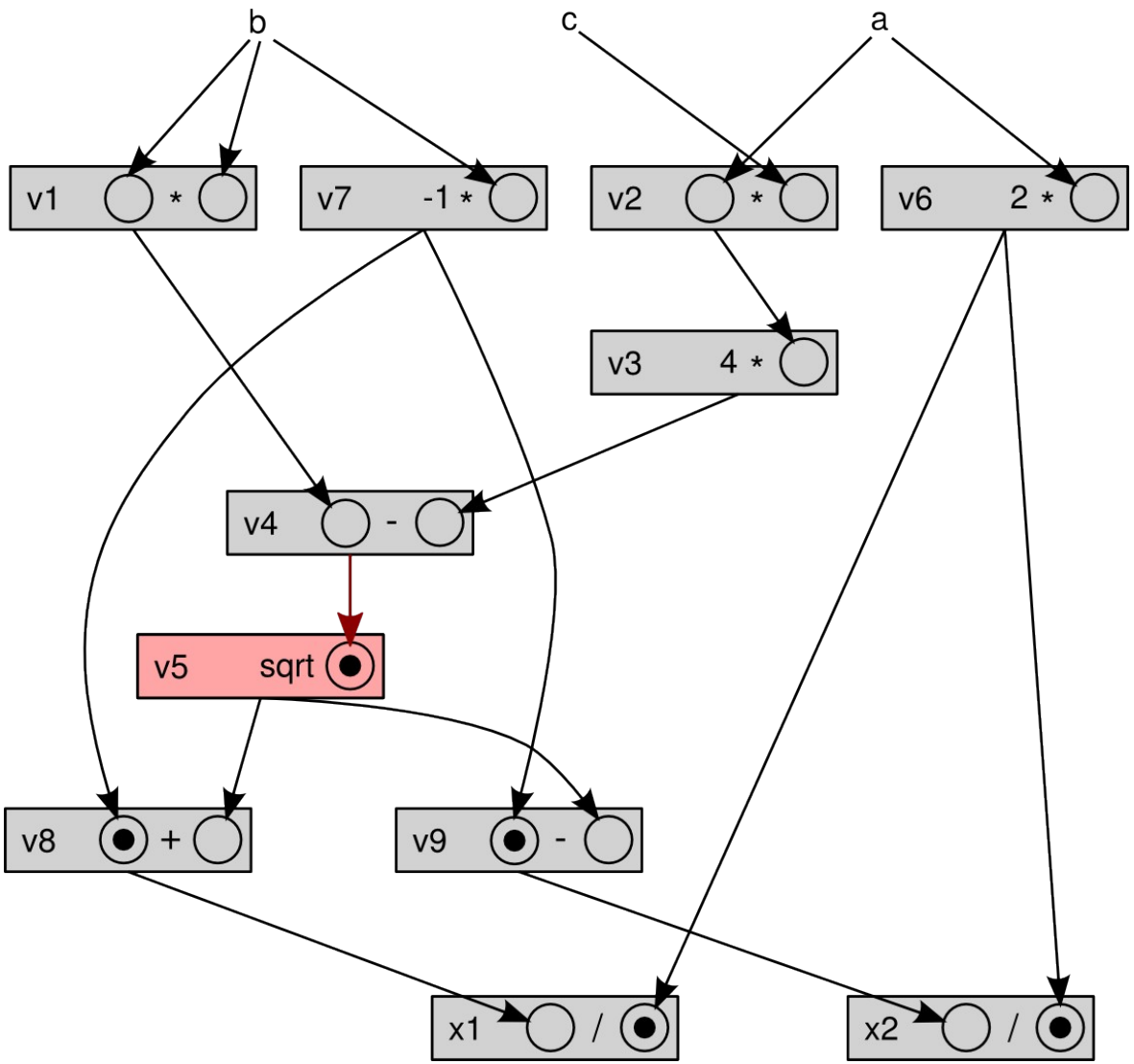
DATA-FLOW MODEL



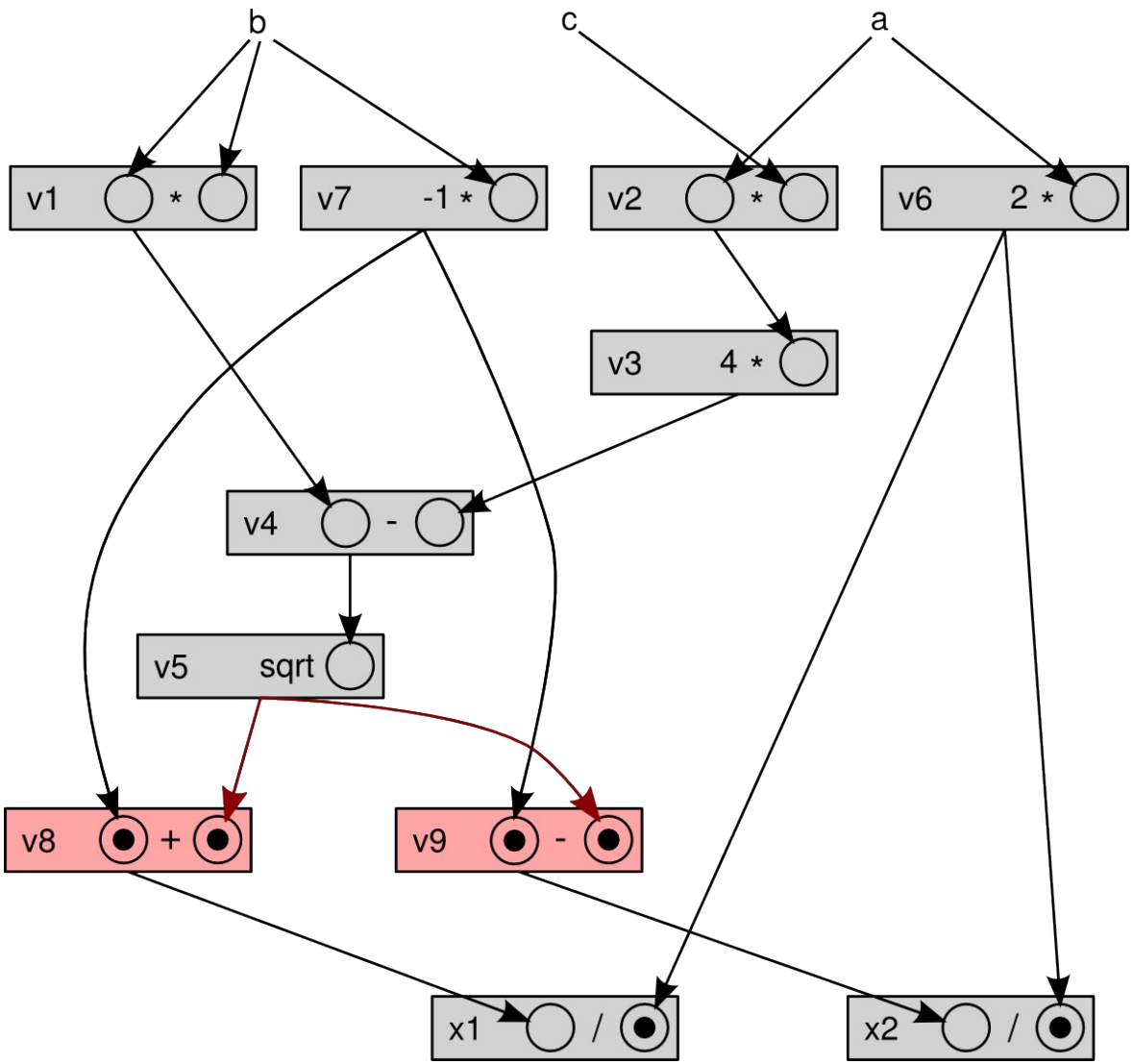
DATA-FLOW MODEL



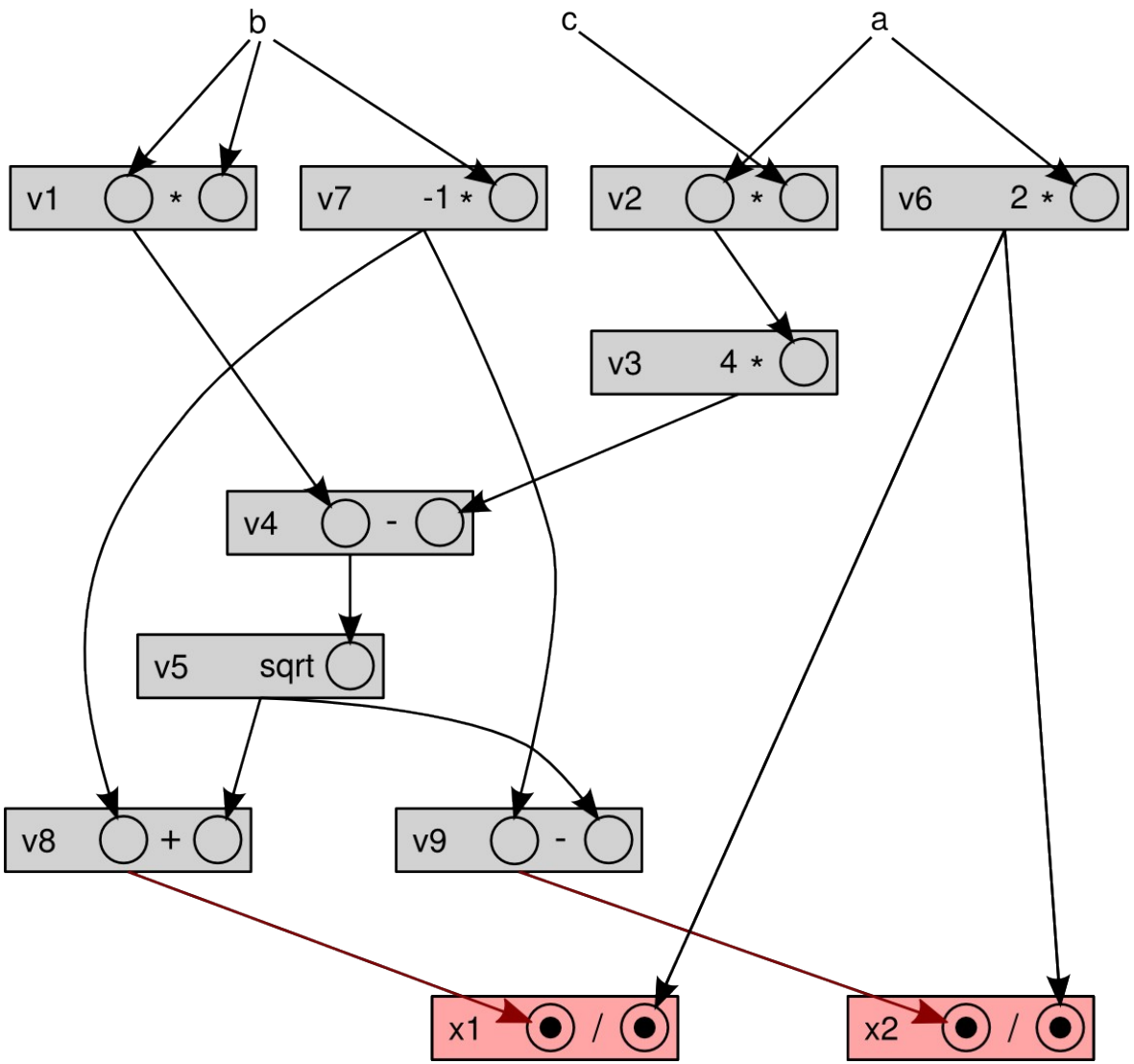
DATA-FLOW MODEL



DATA-FLOW MODEL



DATA-FLOW MODEL



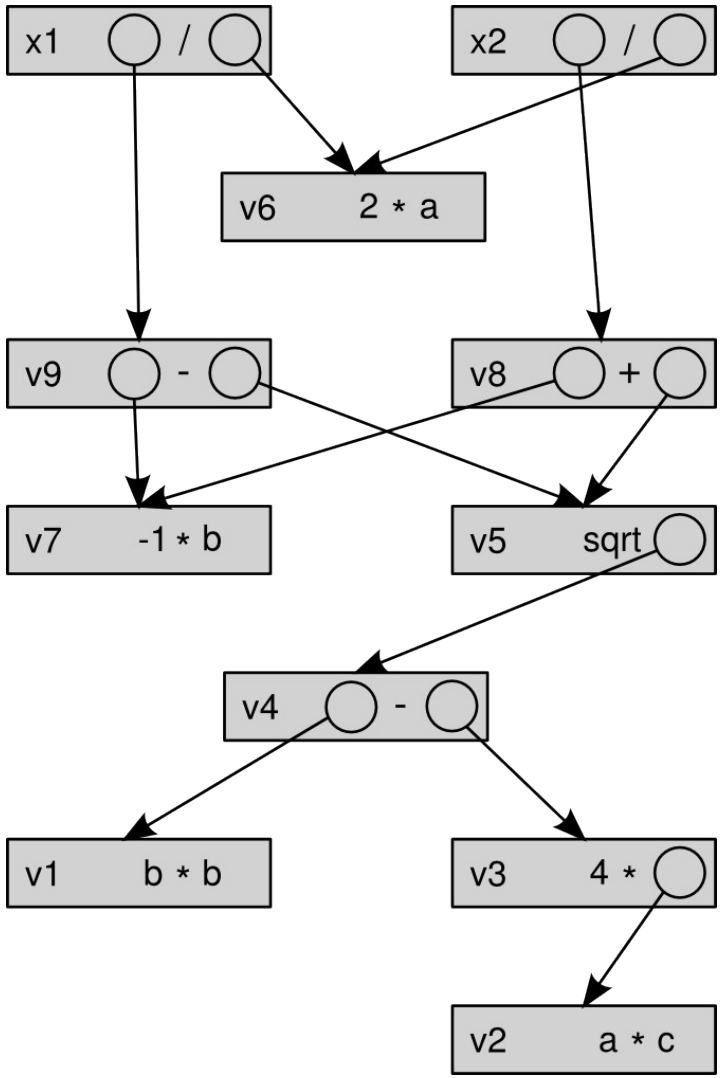
DATA-FLOW MODEL

- Features
  - It is able to detect and utilize parallelism
  - Pure data-flow computers did not become popular
  - But the principle did become popular:
    - Spreadsheets
    - Out-of-order execution of instructions in modern CPUs
    - Data-processing systems (Apache Pig)

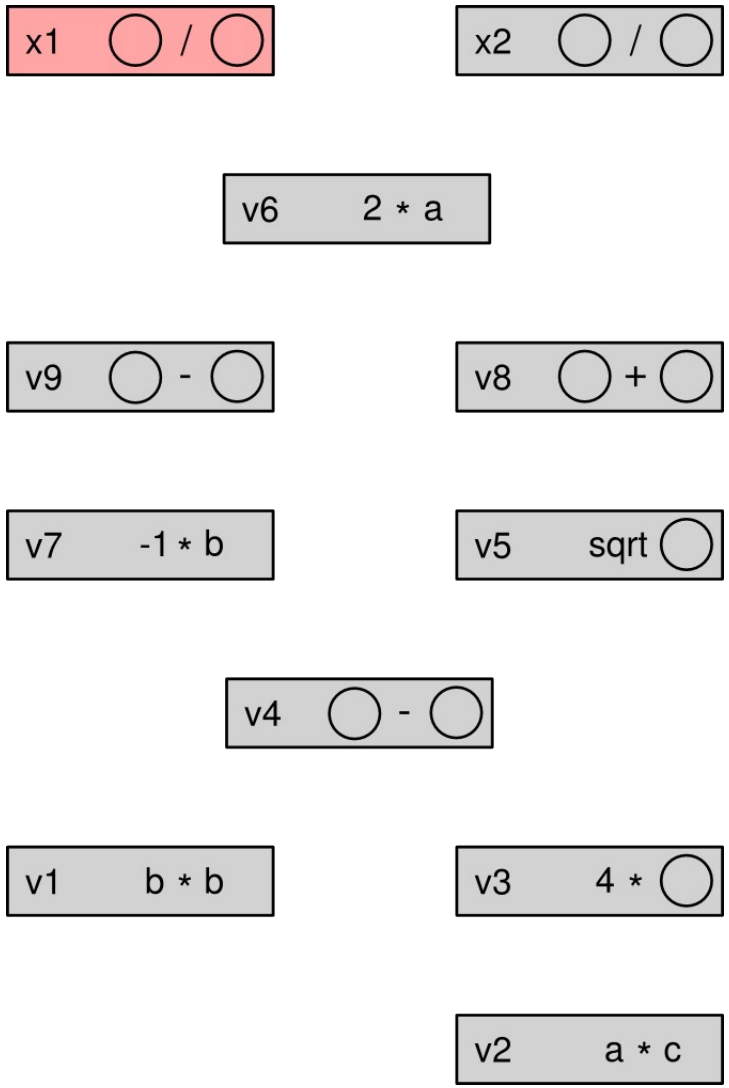
DEMAND DRIVEN MODEL

- Execution of the instructions is automatic:
  - **Every instruction is executed as soon as another instruction needs its result**
- **Comparison**: when instructions are executed
  - Control-flow: when the control token selects it
  - Data-flow: when all operands are available
  - Demand driven: when another instruction needs its result
- Example: solutions of quadratic equations

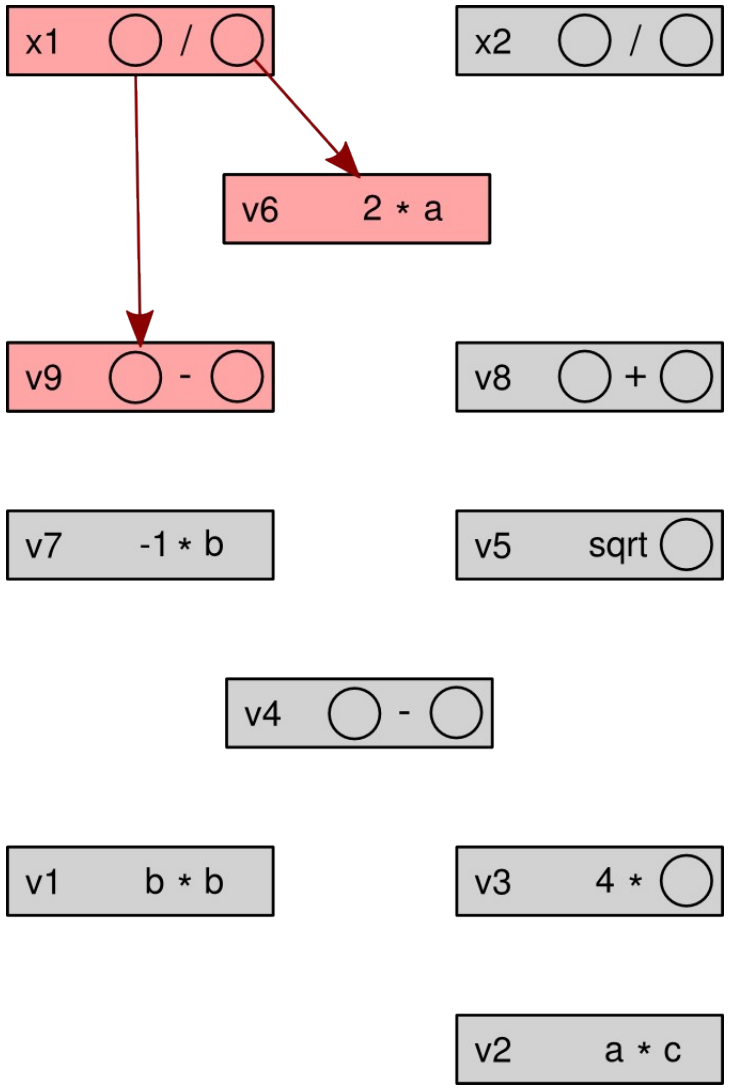
DEMAND DRIVEN MODEL



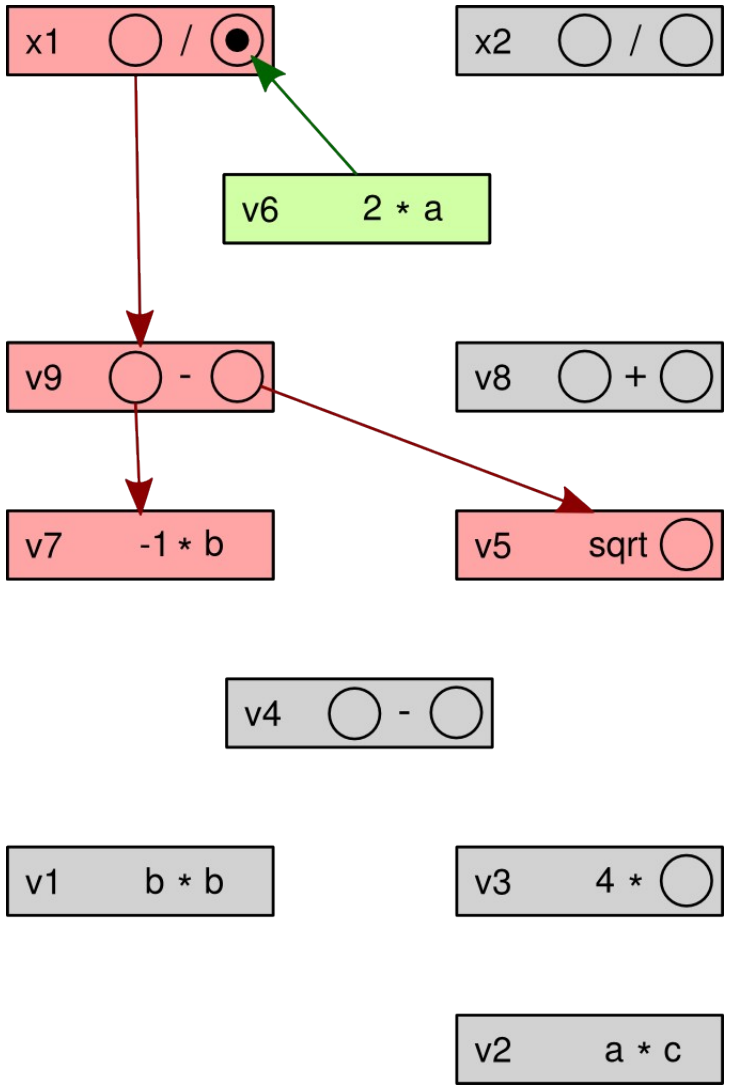
DEMAND DRIVEN MODEL



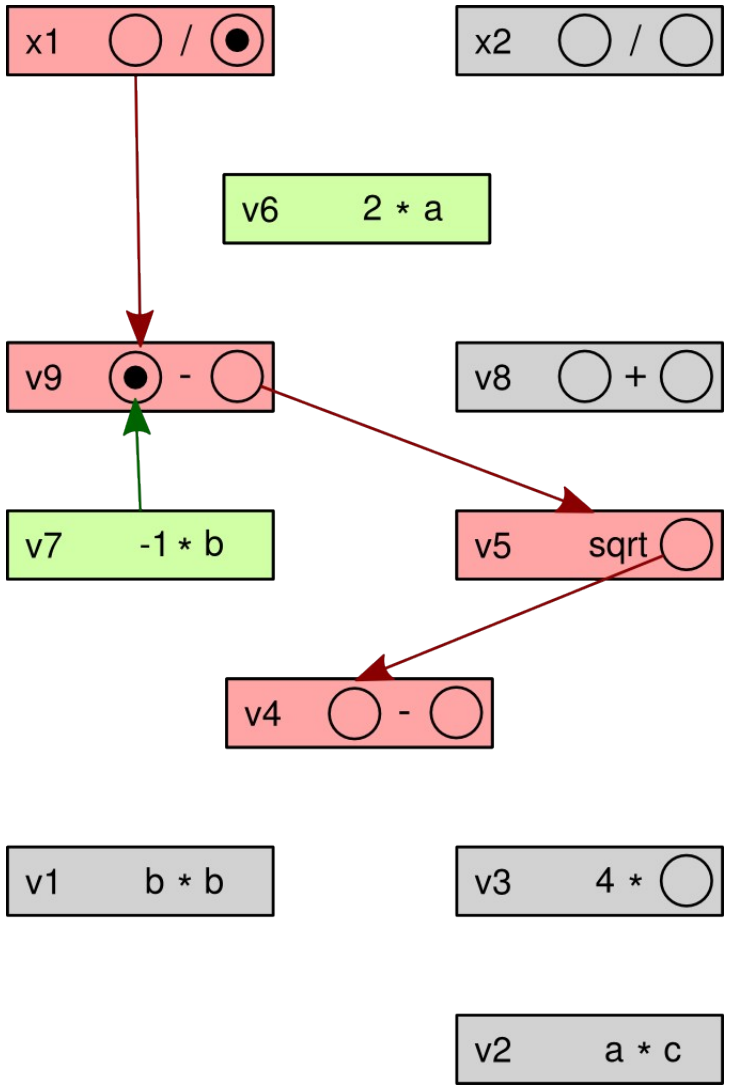
DEMAND DRIVEN MODEL



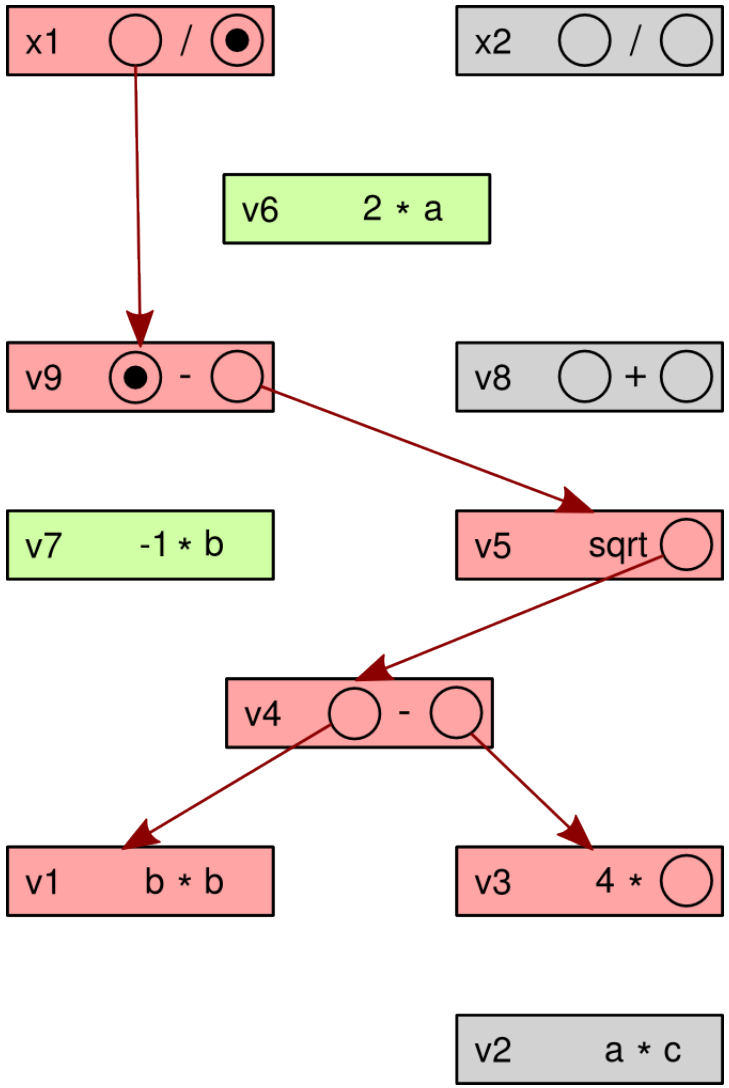
DEMAND DRIVEN MODEL



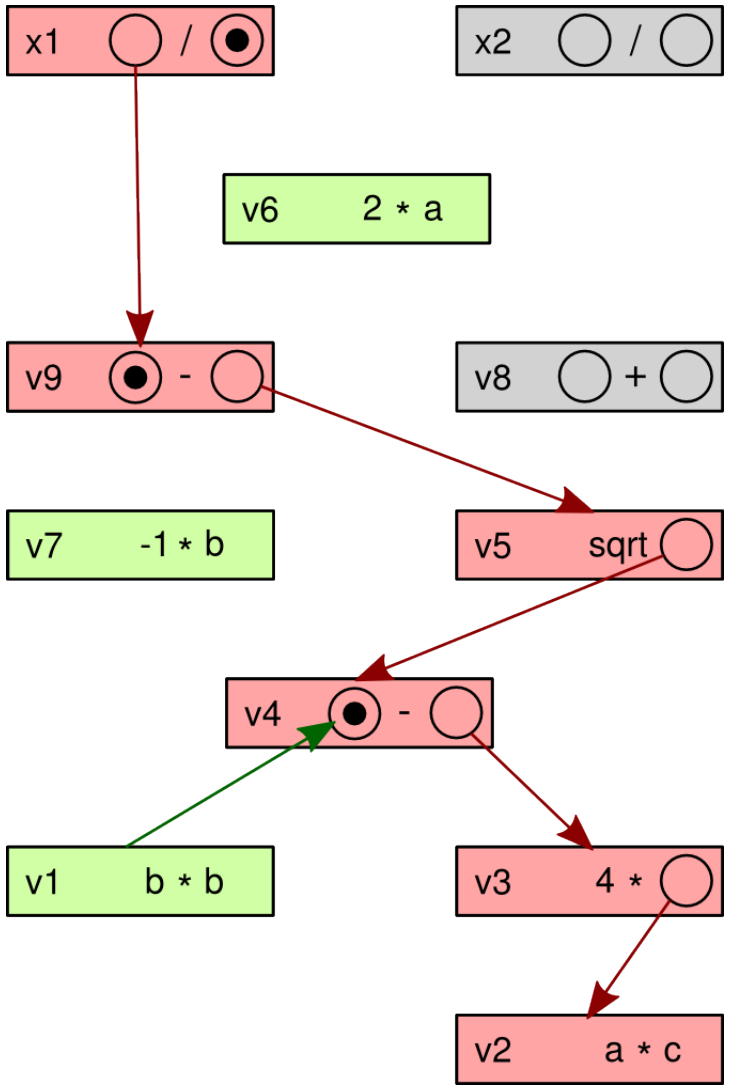
DEMAND DRIVEN MODEL



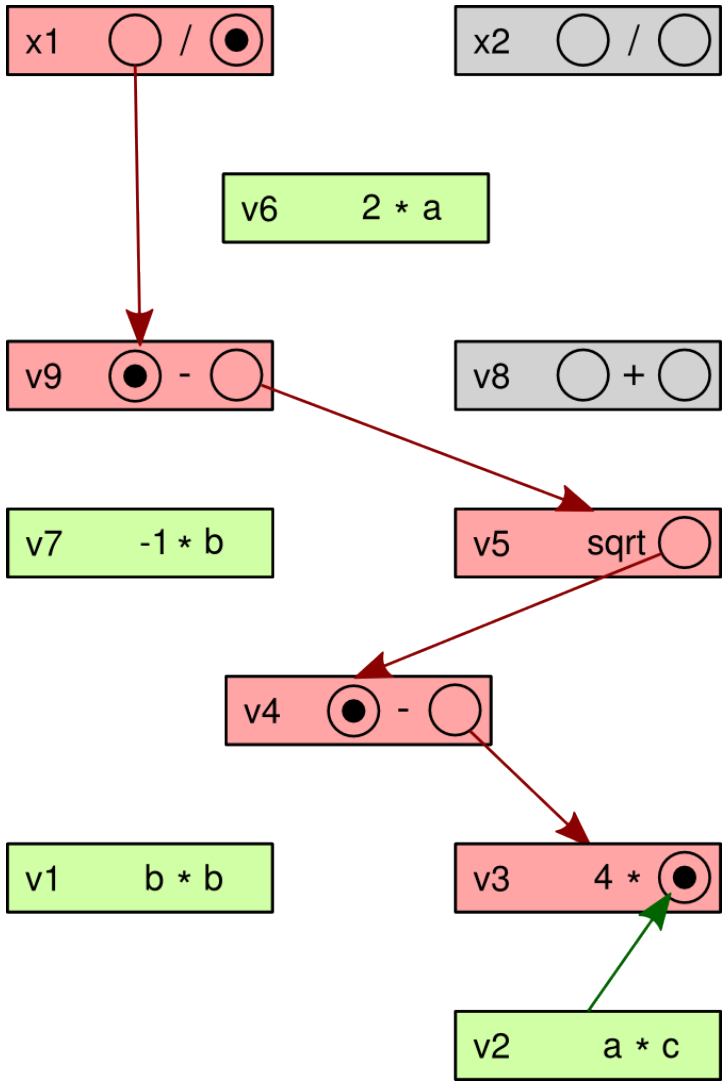
DEMAND DRIVEN MODEL



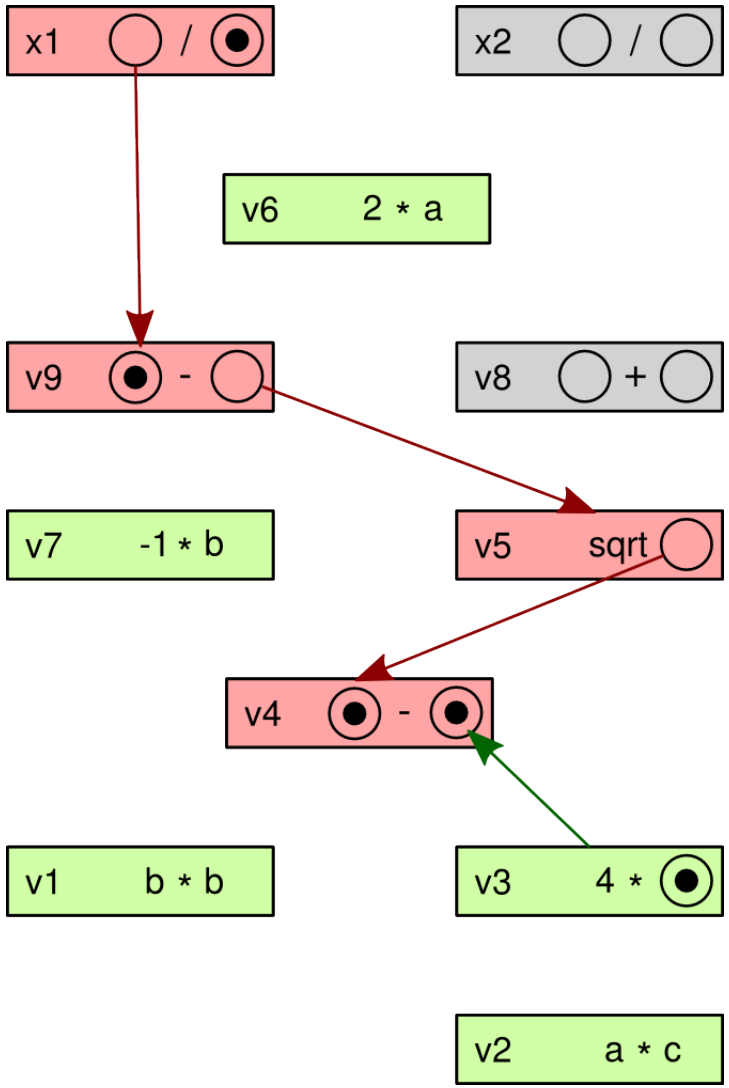
DEMAND DRIVEN MODEL



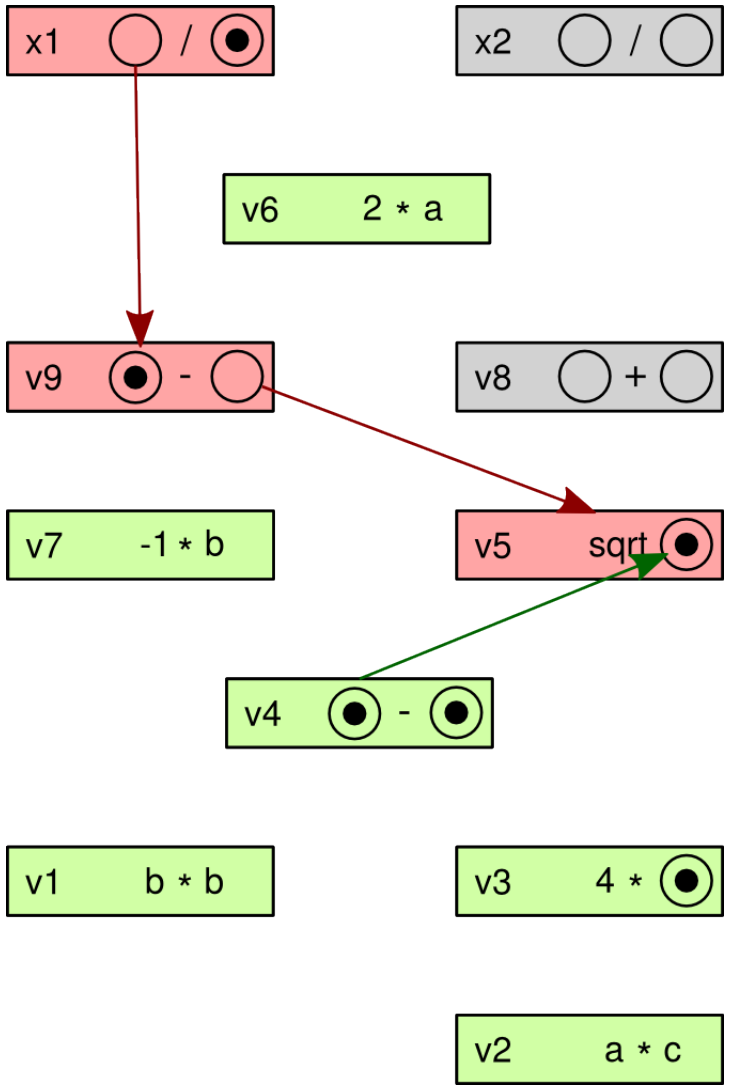
DEMAND DRIVEN MODEL



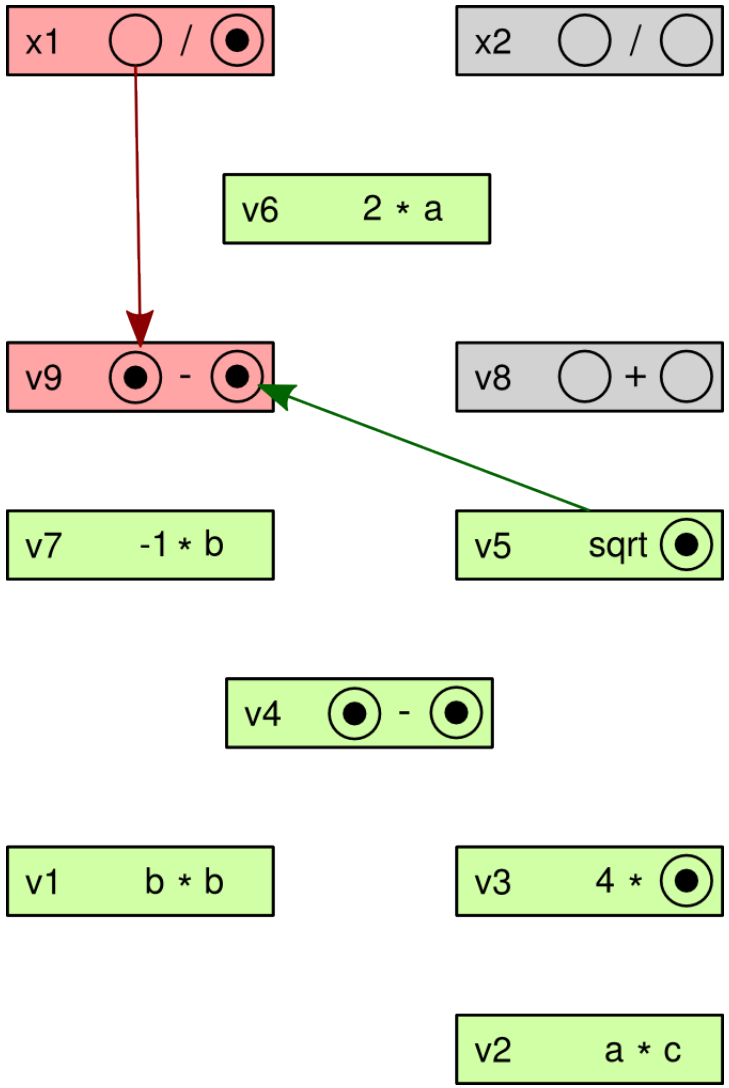
DEMAND DRIVEN MODEL



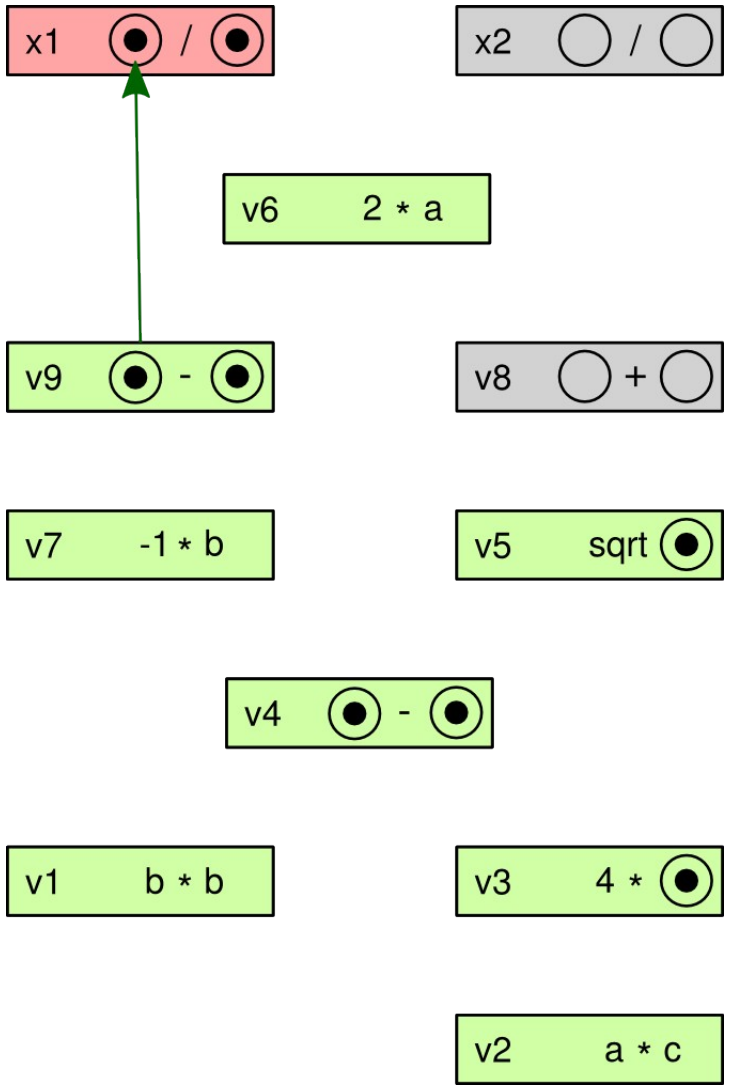
DEMAND DRIVEN MODEL



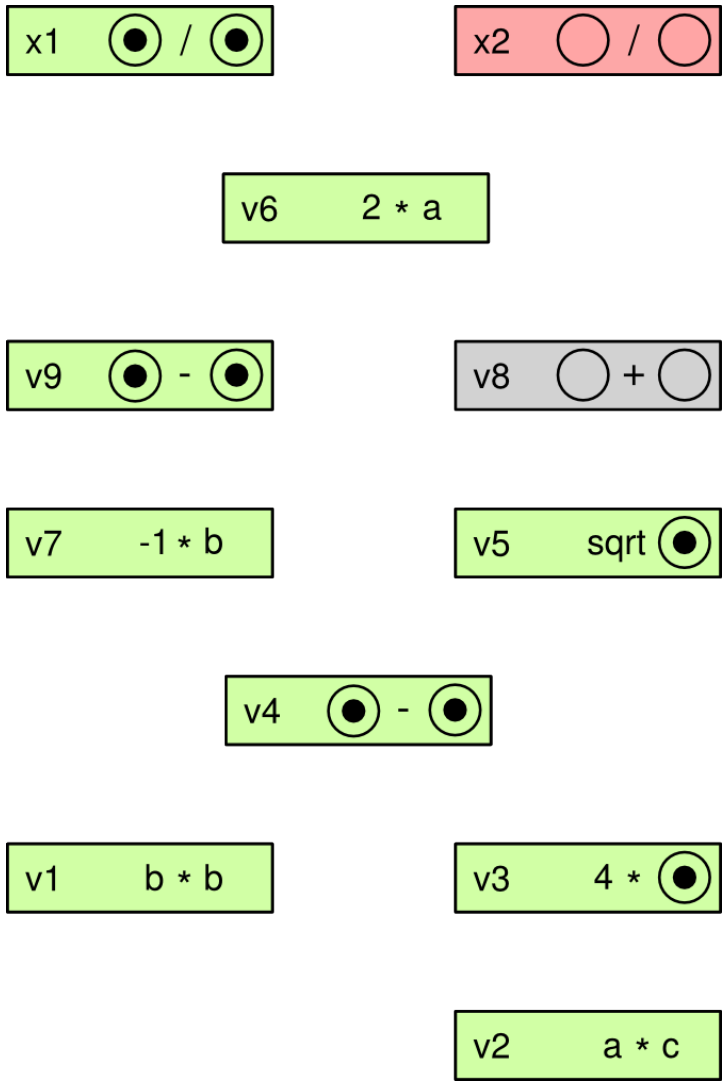
DEMAND DRIVEN MODEL



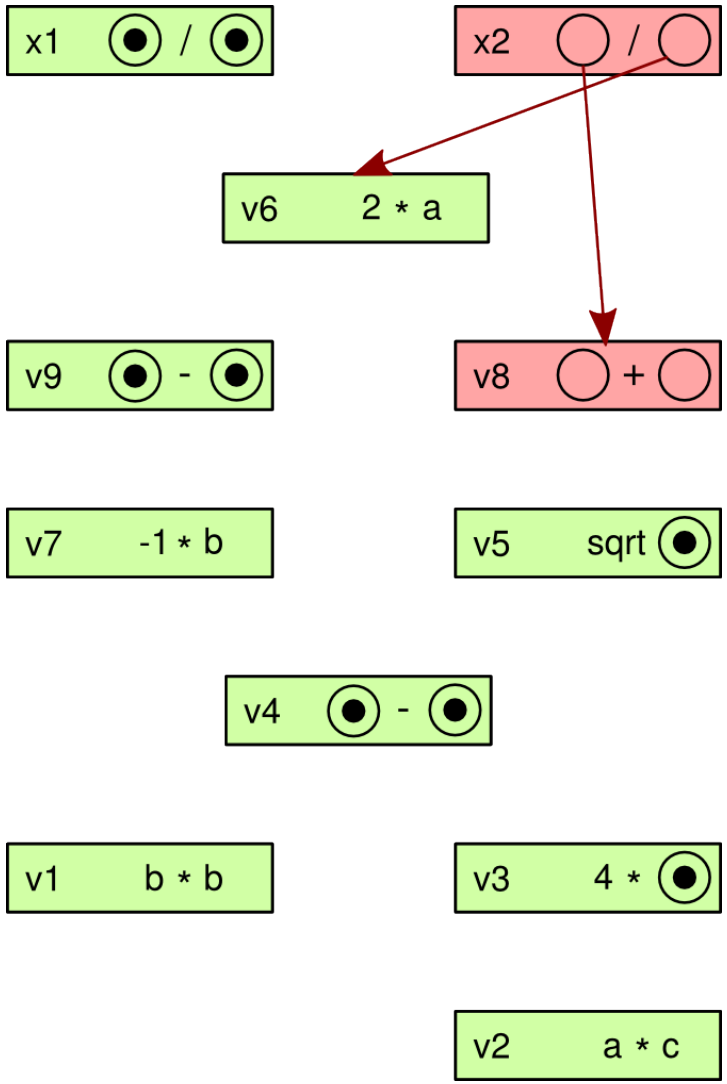
DEMAND DRIVEN MODEL



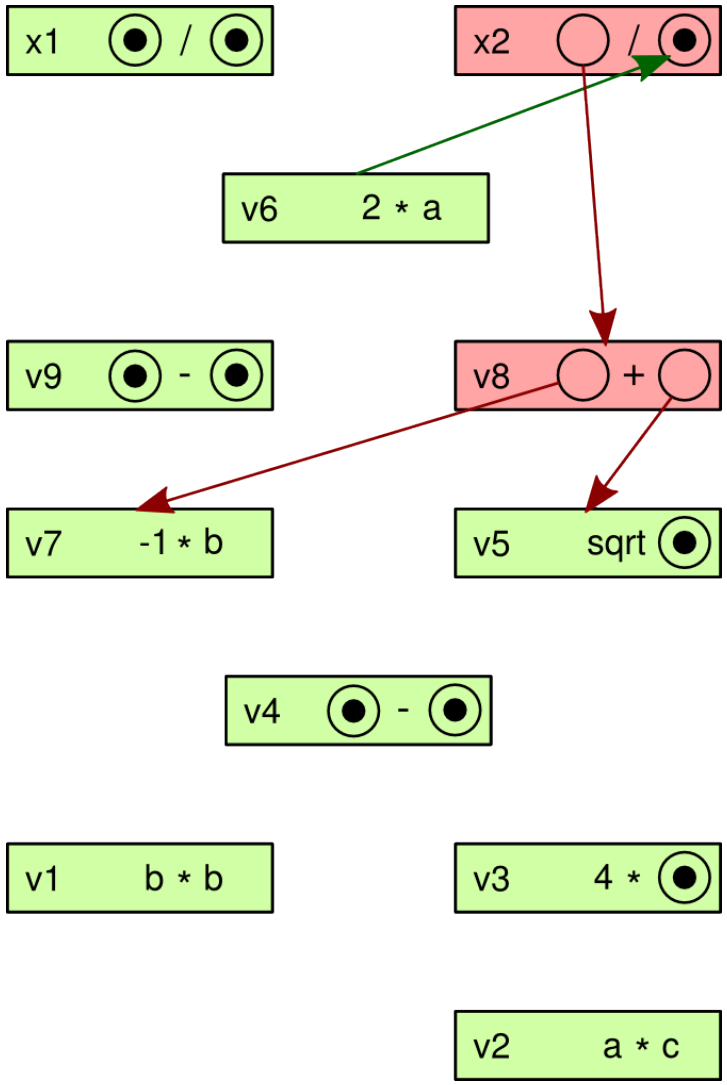
DEMAND DRIVEN MODEL



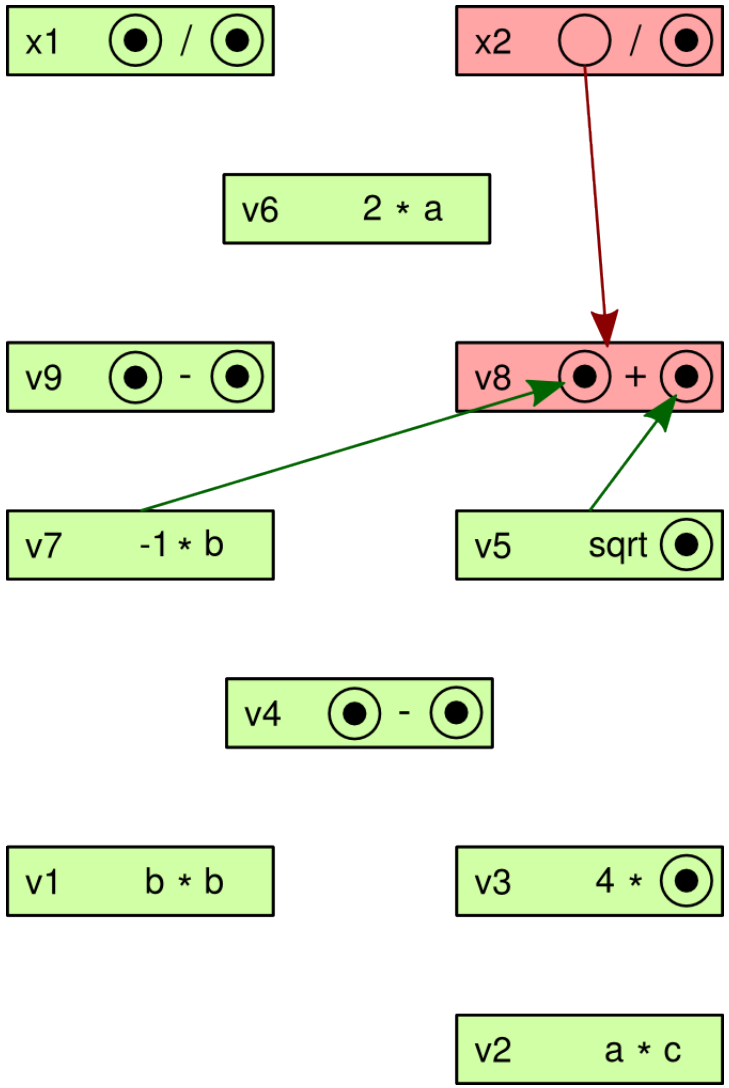
DEMAND DRIVEN MODEL



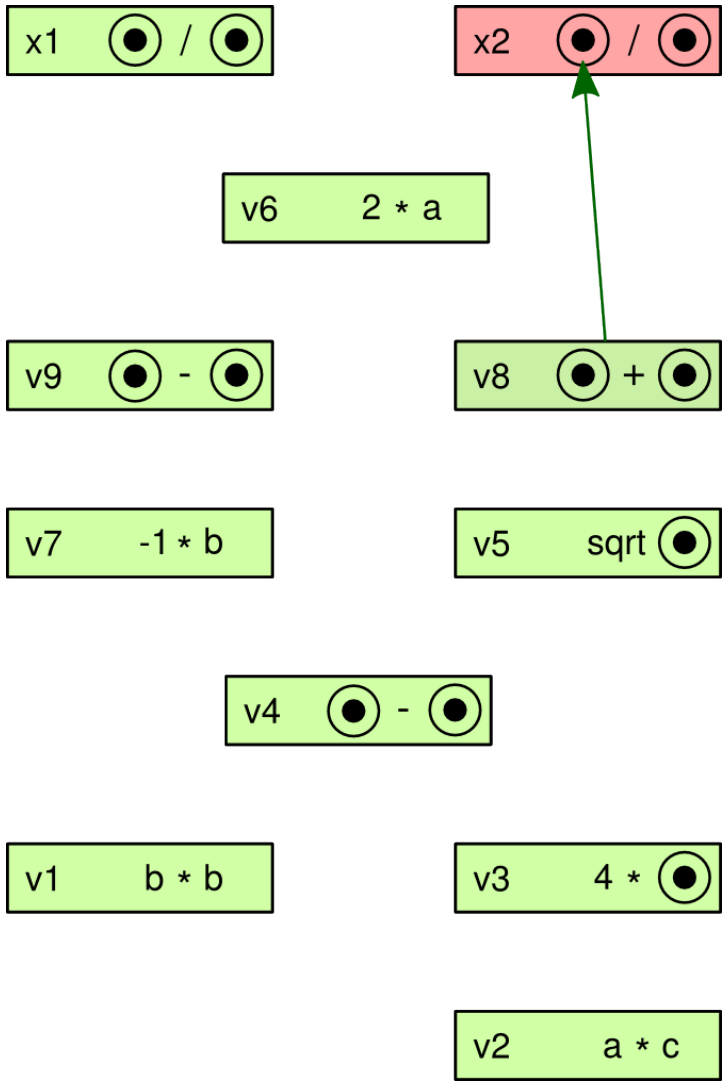
DEMAND DRIVEN MODEL



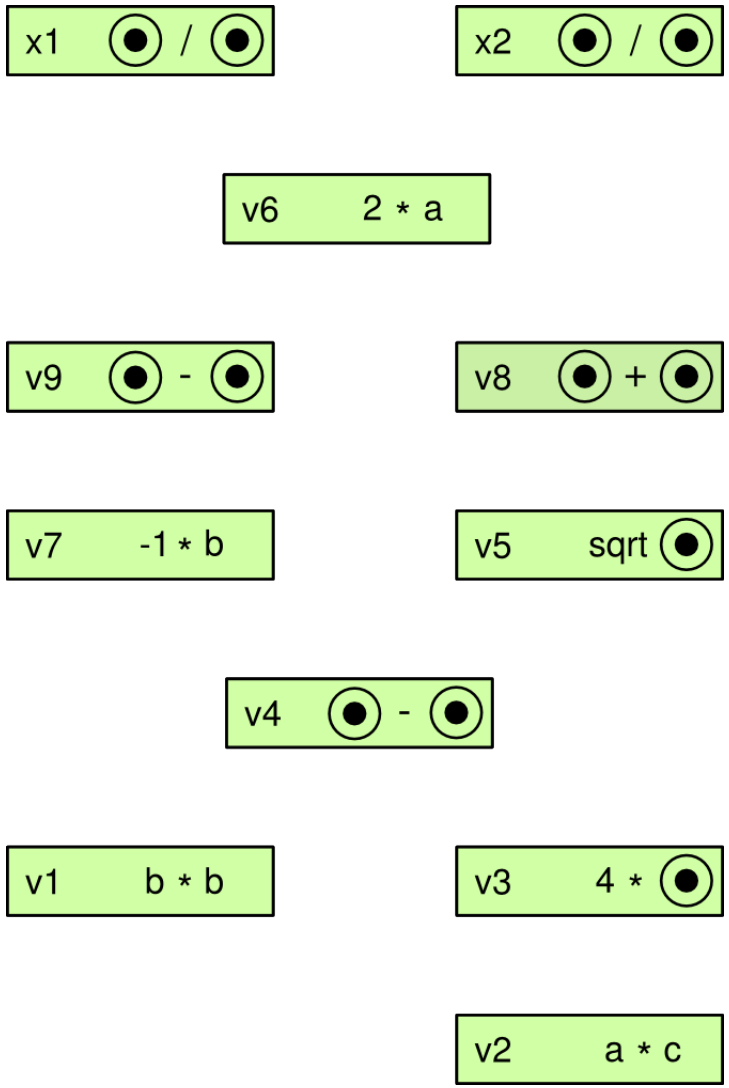
DEMAND DRIVEN MODEL



DEMAND DRIVEN MODEL



DEMAND DRIVEN MODEL



DEMAND DRIVEN MODEL

- Features
  - It is able to detect and utilize parallelism
  - Pure demand driven computers did not become popular
  - But the principle did become popular:
    - Functional programming languages
    - Google MAP-REDUCE paradigm

INFORMATION PROCESSING MODELS

- Summary:
  - The programming interface of most CPUs follows the control-flow model
  - Functional languages allow demand-driven programming over CPUs following the control-flow model
  - CPUs following the control-flow model may use data-flow model based re-ordering of the instructions internally

Control-driven architectures

THE NEUMANN ARCHITECTURE

- „Official story”:
  - 1945, "First Draft of a Report on the EDVAC"
  - First computer following a Neumann architecture: Institute for Advanced Study, Princeton University (1945-1951)
- The truth:
  - Turing – 1936: first paper describing the concept, but computer scientists were not aware of it
  - Eckert & Mauchly – 1943: Neumann wrote a paper based on the ideas of Eckert & Mauchly. His colleague started to circulate it with only Neumann's name on it.

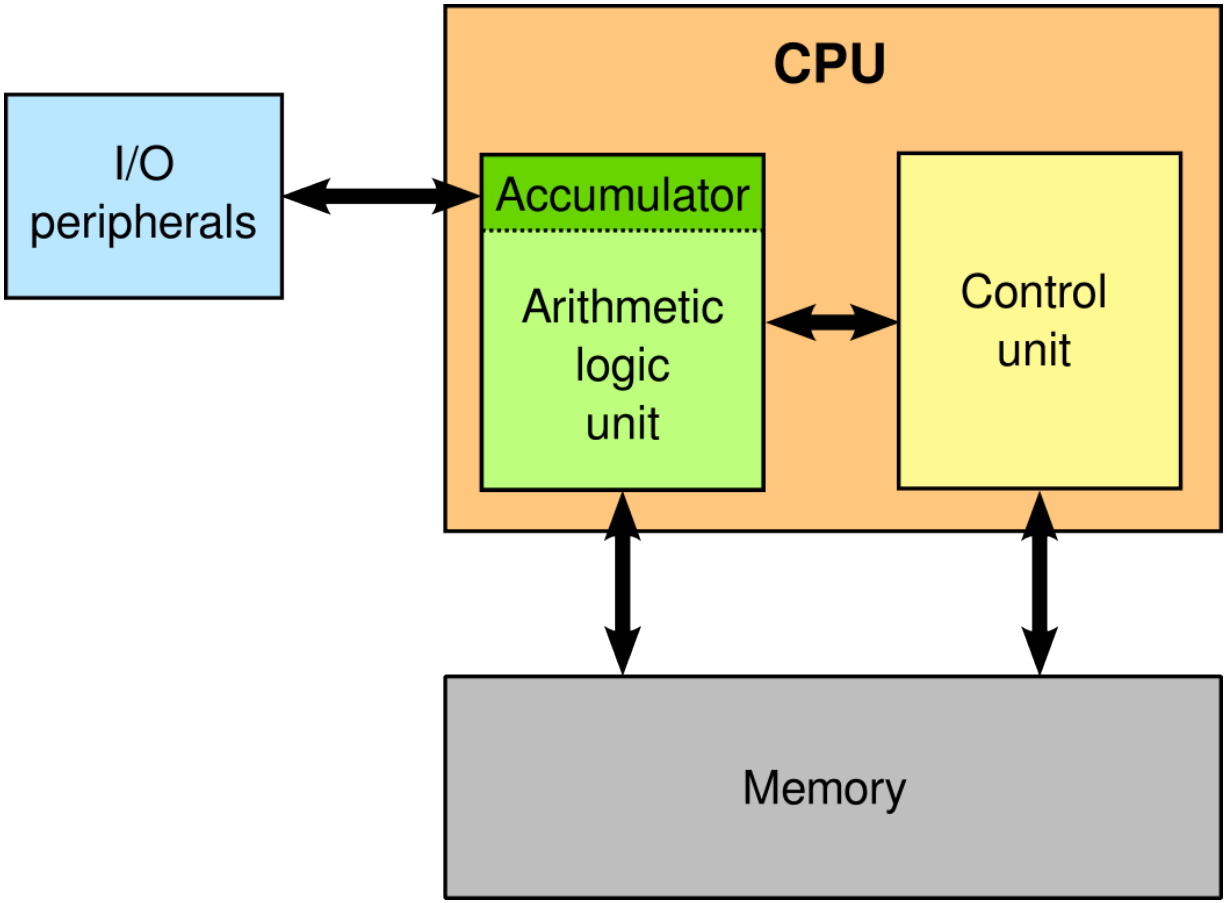
THE NEUMANN ARCHITECTURE

- Novelty:

The instructions are stored in the memory together with the data

  - Earlier (before Neumann):
    programming = re-building or re-wiring the computer
  - Main motivation behind the Neumann architecture:
    - Easier to change the program
      (A new program for a non-Neumann machine like ENIAC took 3 weeks to set up)
    - It is easier to run several programs one after the other
    - It made it possible to write programs generating a program
      - The birth of compilers

THE NEUMANN ARCHITECTURE

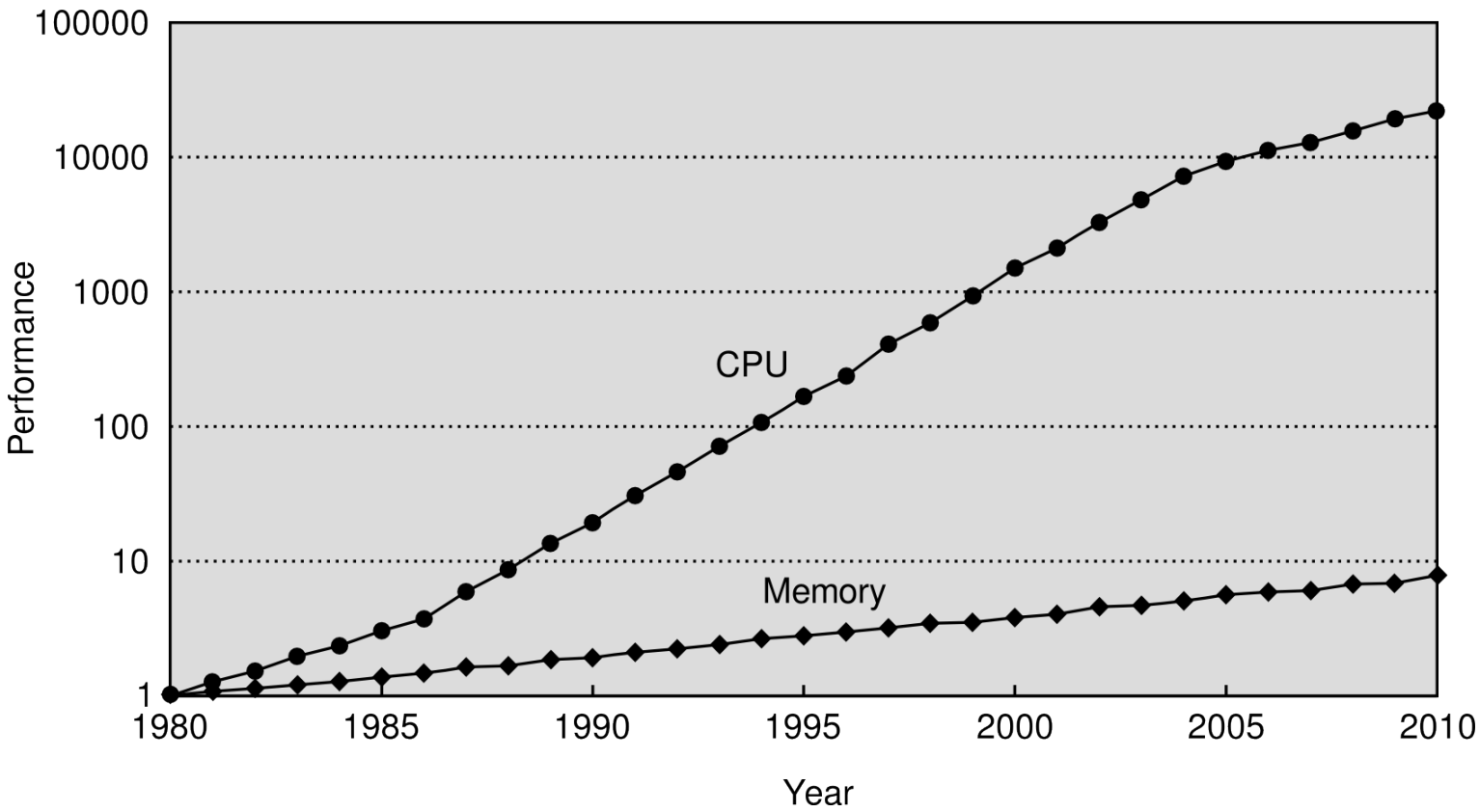


THE NEUMANN ARCHITECTURE

- Content of the memory:
  - Data units of fixed size
  - Can be:
    - Instruction
    - Integer number
    - Character
    - Floating point number
    - …
  - How do we know what is stored in the memory at a given address?
  - **We do not know it!**
  - It depends on what kind of instruction is accessing it

THE NEUMANN ARCHITECTURE

- Bottleneck:
  - **Memory bandwidth!**



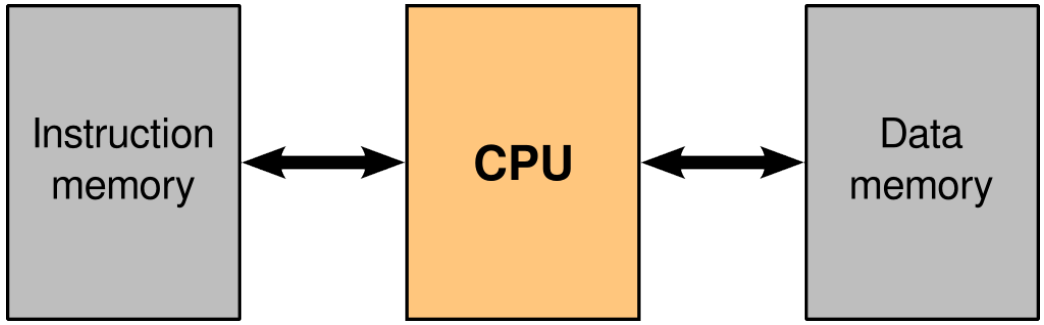
THE NEUMANN ARCHITECTURE

- Self-modifying programs
  - Replace their own instructions by memory operations
  - Some algorithms can be shorter this way
  - Self modifying programs are impossible to maintain
  - Never do it!

THE HARVARD ARCHITECTURE

- Harvard Mark I:
  - Reads the instructions from a punched card
  - Data is stored by a memory consisting of relays
- Principally different from the Neumann architecture :

The instructions and the data are stored separately

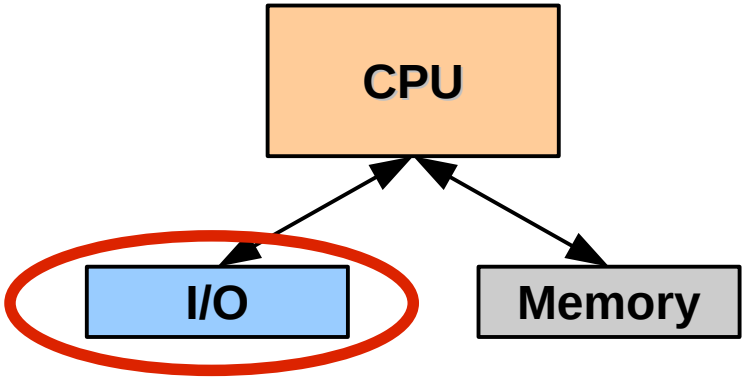


THE HARVARD ARCHITECTURE

- The classical Harvard architecture
  - Instructions: exclusively in the instruction memory
  - Data: exclusively in the data memory
- The two kinds on memory accesses can be overlapped:
  - While fetching the operands of the i-th instruction from the data memory…
  - ...the processor can fetch the (i+1)-th instruction from the instruction memory at the same time
- Modified Harvard architecture:
  - The instruction memory can be read by the program as well
  - It can store constant data
- Typical applications:
  - Micro-controllers, digital signal processors (PIC, Atmel, ...)

Topics

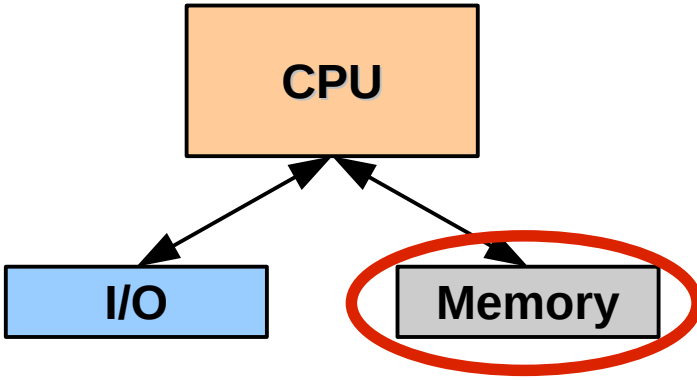
TOPICS



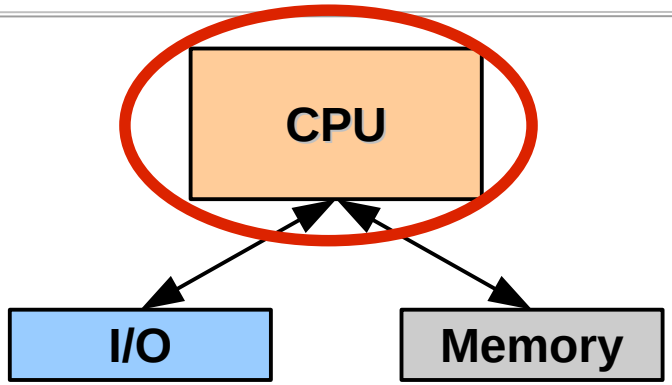
- **I/O peripherals**
  - Lectures
    - Flow control, decreasing the load of CPU, busses, south bridge – north bridge, etc.
    - Mass storage drives: HDD/SSD
    - Interfaces: PCI, PCI Express, USB
    - RAID
  - Classroom practices
    - Project for reviewing digital design:
      - A CPU based door lock project
    - Numerical examples related to I/O peripherals
      - Computing load induced by I/O peripherals
      - HDD delay and throughput calculations
      - SSD block management and garbage collection algorithms

TOPICS

- **Memory**
  - Lectures
    - Memory technologies:
      SRAM, DRAM, SDRAM, DDR, DDR2, DDR3, DDR4, DDR5, GDDR3, GDDR5, etc., memory timings
    - Virtual memory system
    - Cache memory
    - Locality aware programming techniques
  - Classroom practice
    - Numerical examples
      - Scheduling DRAM commands
      - Page table management
      - Cache management



TOPICS



- **CPU**
  - Lecture
    - Pipeline instruction processing
    - Out-of-order instruction scheduling
    - Superscalar processors
    - Wide pipelines
    - Branch prediction
  - Classroom practice
    - Numerical examples:
      - Instruction scheduling, identifying hazards and dependencies
      - Optimizing the order of instructions

TOPICS

- **Parallel systems**
  - SIMD processing
    - Vector processors
    - Vector instruction sets
    - Vectorization
  - Multiprocessor systems
    - Multi threaded and multi-core systems
    - Multiprocessor systems
    - Interconnections
    - Shared memory management

DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

MŰEGYETEM 1782