The Von Neumann Model

Equivalence between hardware and software: Anything that can be done in software can be done in hardware, and vice-versa.

Breakthrough: Instead of programming the machines by changing the wiring, store a program in memory (but this had to wait until suitable storage means were available)

Components:

CPU – includes ALU, control unit, registers, program counter

Main memory system

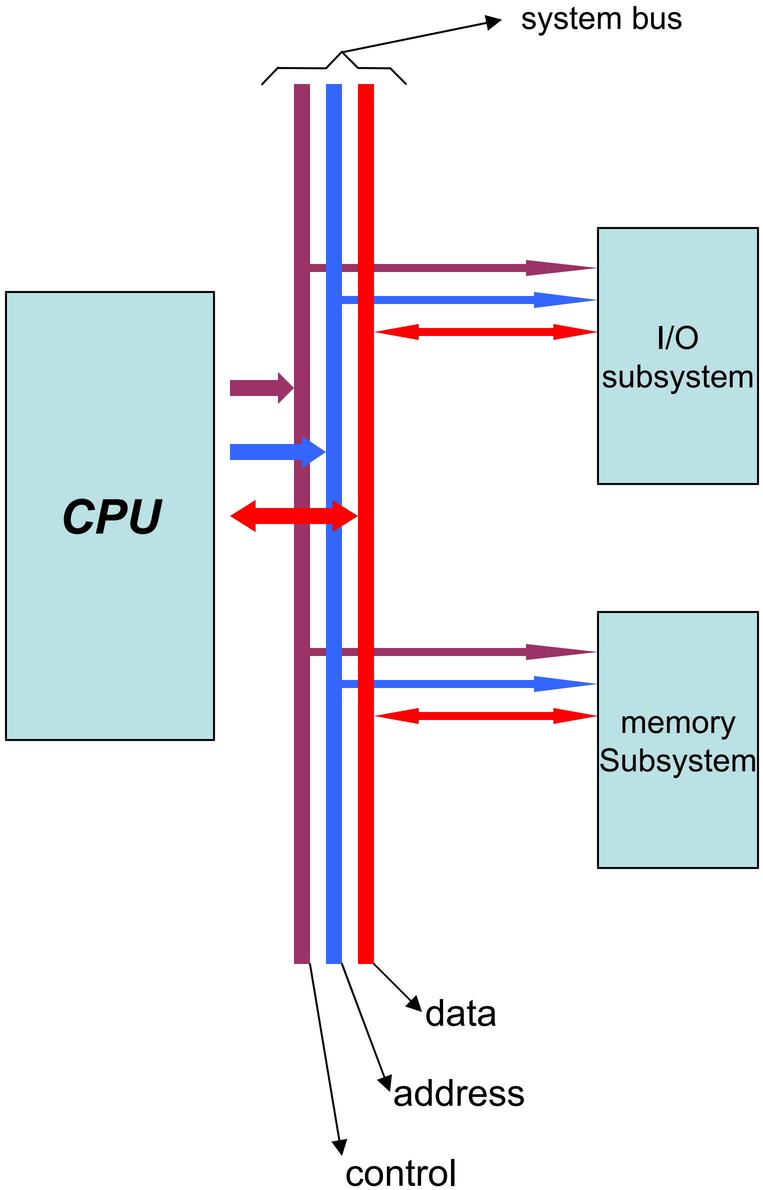
I/O subsystem to connect external devices

Fundamental properties:

Sequential instruction processing

Alternates between instruction fetching and execution (more on this soon)

The Von Neumann Model *(note: this was proposed earlier by John Mauchly and Presper Eckert – creators of ENIAC)*



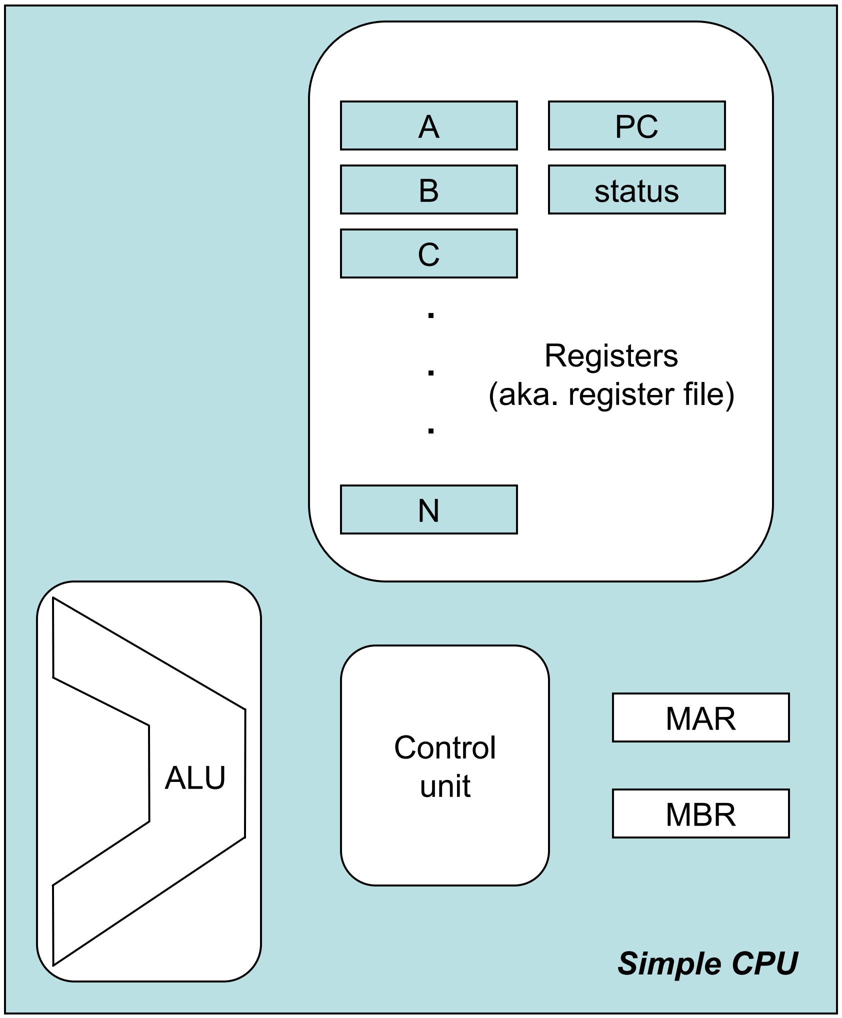
Data bus – transfer information between the CPU and the memory and peripherals

Address bus – provides the address information required to access specific memory locations or peripheral devices

Control bus – provides control signals to memory and devices

The Von Neumann Model

A block diagram of a simple CPU



ALU: Arithmetic and Logic Unit: does all the number crunching and logic operations, computes addresses, executes instructions

Registers: Fastest memory in the system, they store data and results for the ALU. Special purpose registers: ***PC (program counter)*** keeps address of the next instruction to be executed, ***status*** register contains flags that indicate special conditions resulting from the execution of instructions (e.g. overflow, divide by zero)

Control unit: Takes care of instruction fetch and decoding, Interfacing with the memory system, and general sequencing of operations within the CPU

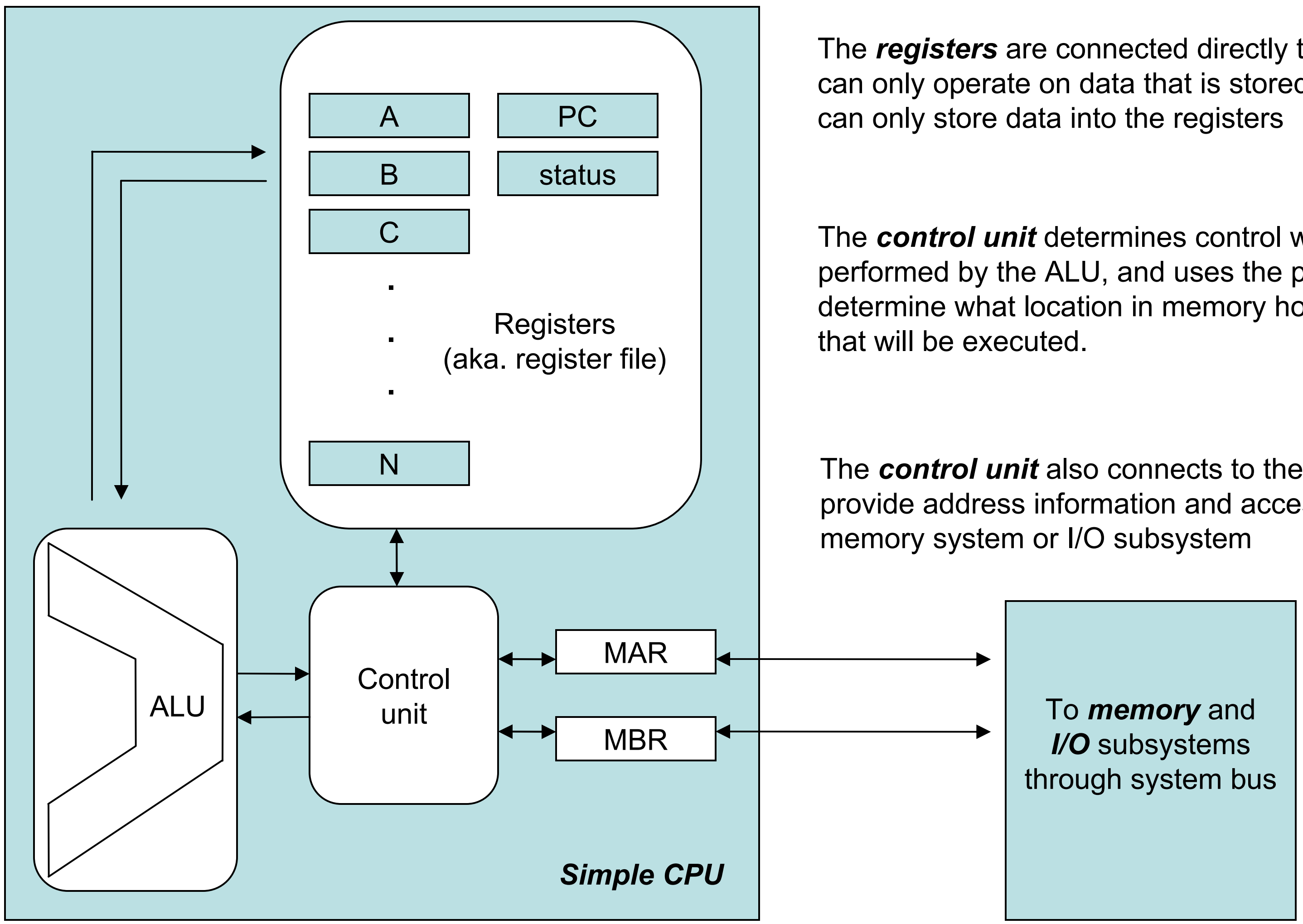
MAR: Memory address register. Contains the address of whatever memory location is being currently referenced

MBR: Memory buffer register. Contains the data being read from, or written to memory

Each CPU will have their own special purpose registers, this is just a general model

The Von Neumann Model

A block diagram of a simple CPU



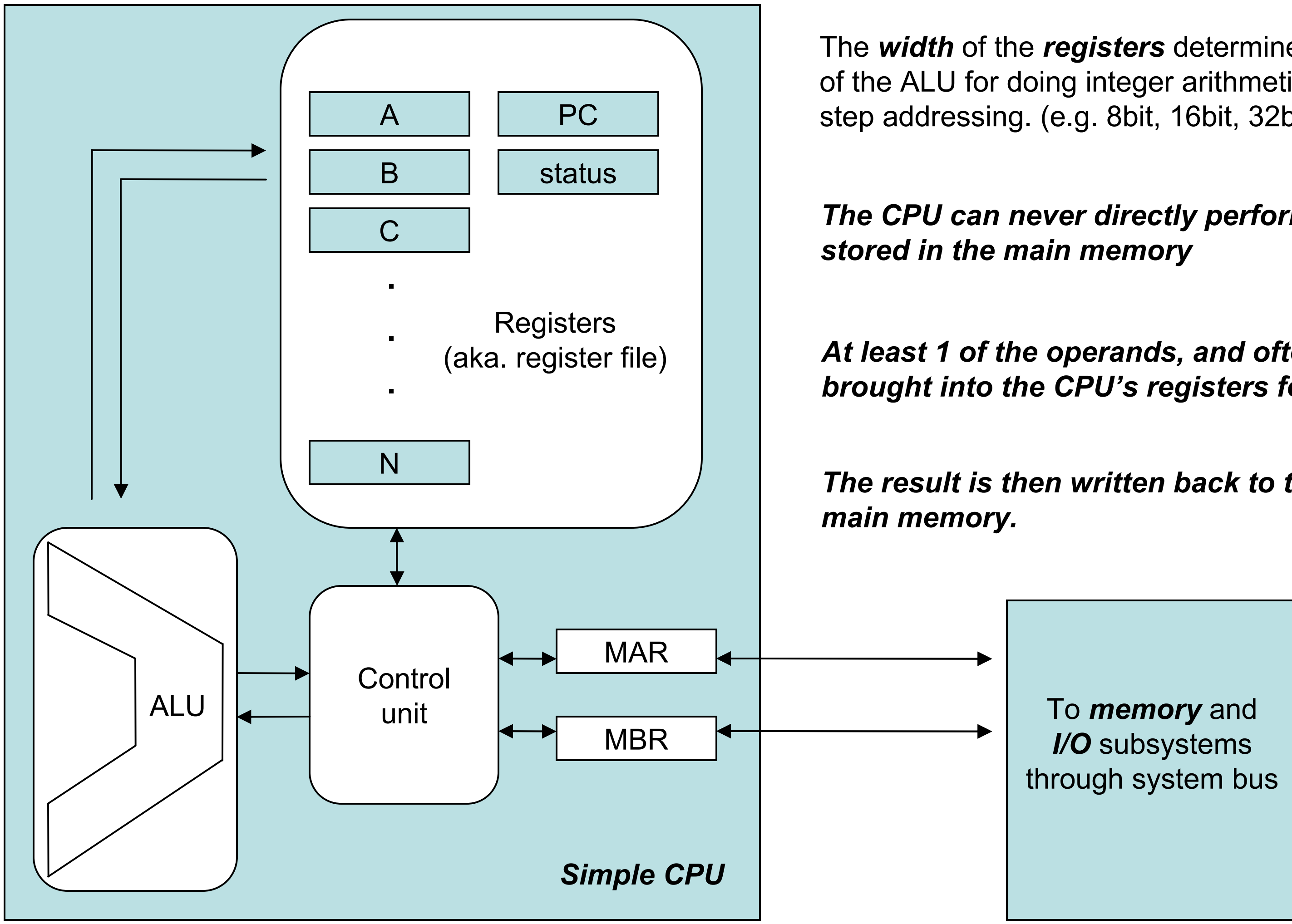
The ***registers*** are connected directly to the ALU. The ALU can only operate on data that is stored in the registers, and can only store data into the registers

The ***control unit*** determines control what operations are performed by the ALU, and uses the program counter to determine what location in memory holds the next instruction that will be executed.

The ***control unit*** also connects to the ***MAR*** and ***MBR*** to provide address information and access data from the memory system or I/O subsystem

The Von Neumann Model

A block diagram of a simple CPU



The ***width*** of the ***registers*** determines the maximum capacity of the ALU for doing integer arithmetic, logic, and single step addressing. (e.g. 8bit, 16bit, 32bit, 64bit, etc CPUs)

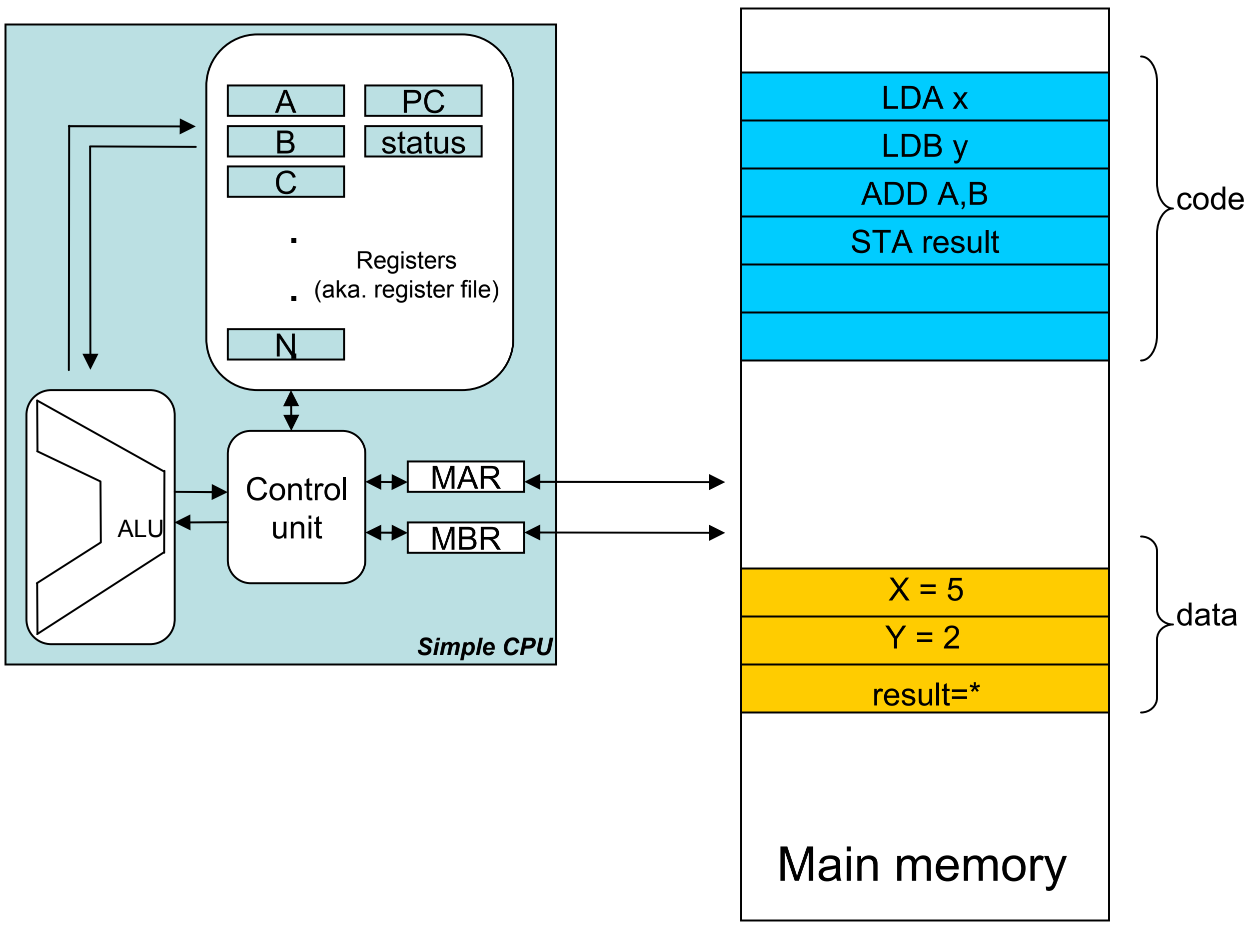
The CPU can never directly perform operations on data stored in the main memory

At least 1 of the operands, and often both, have to be brought into the CPU's registers for manipulation.

The result is then written back to the desired location in main memory.

The Von Neumann Model

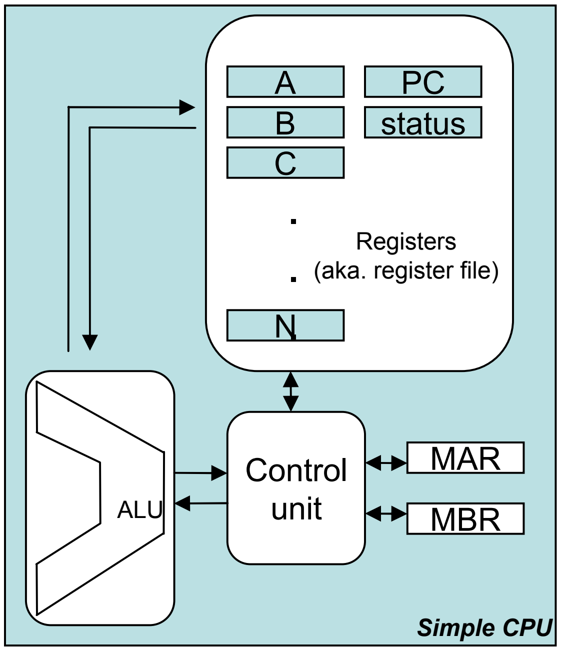
The fetch-decode-execute cycle

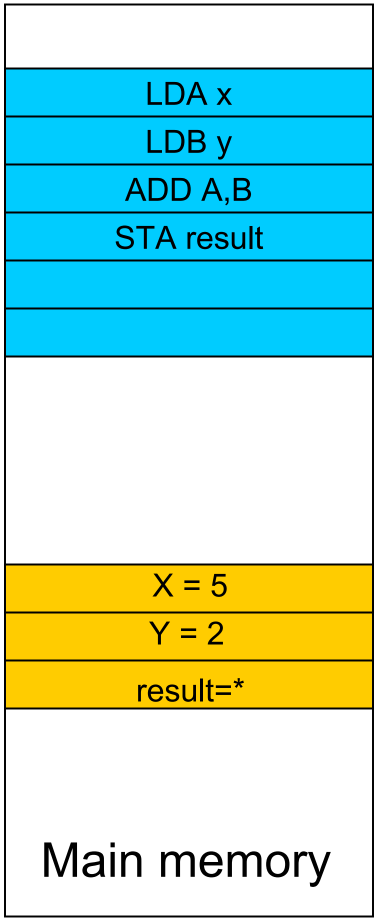


Code and data now stored in main memory

The Von Neumann Model

The fetch-decode-execute cycle

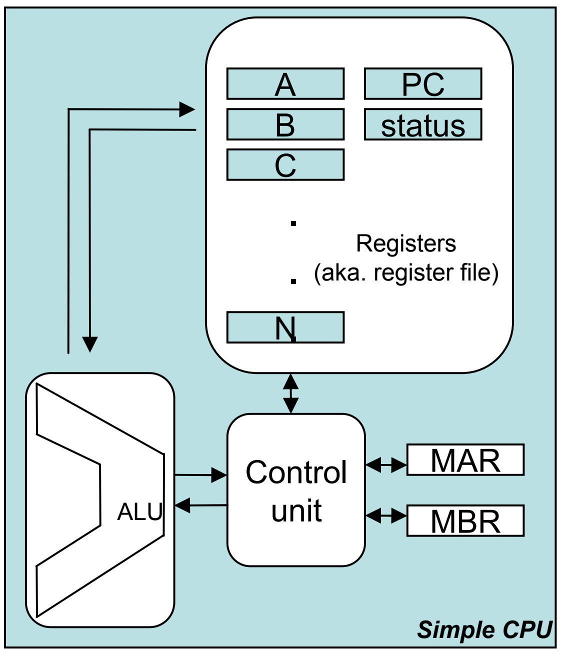


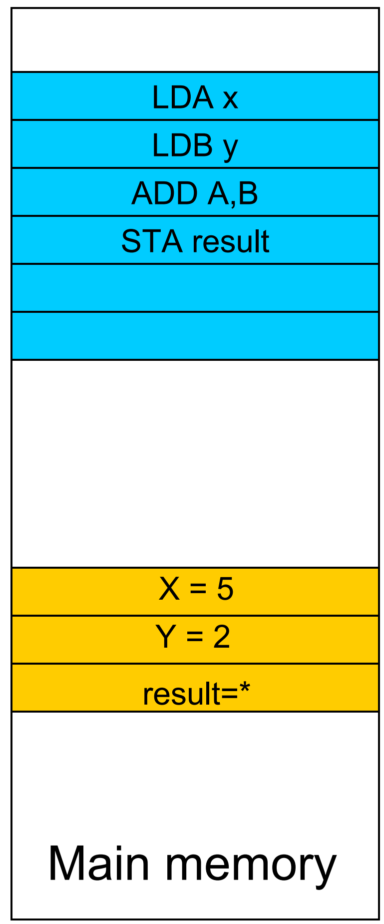


Processor has to alternate between reading code and accessing/manipulating data

The Von Neumann Model

The fetch-decode-execute cycle





3 Stage Cycle:

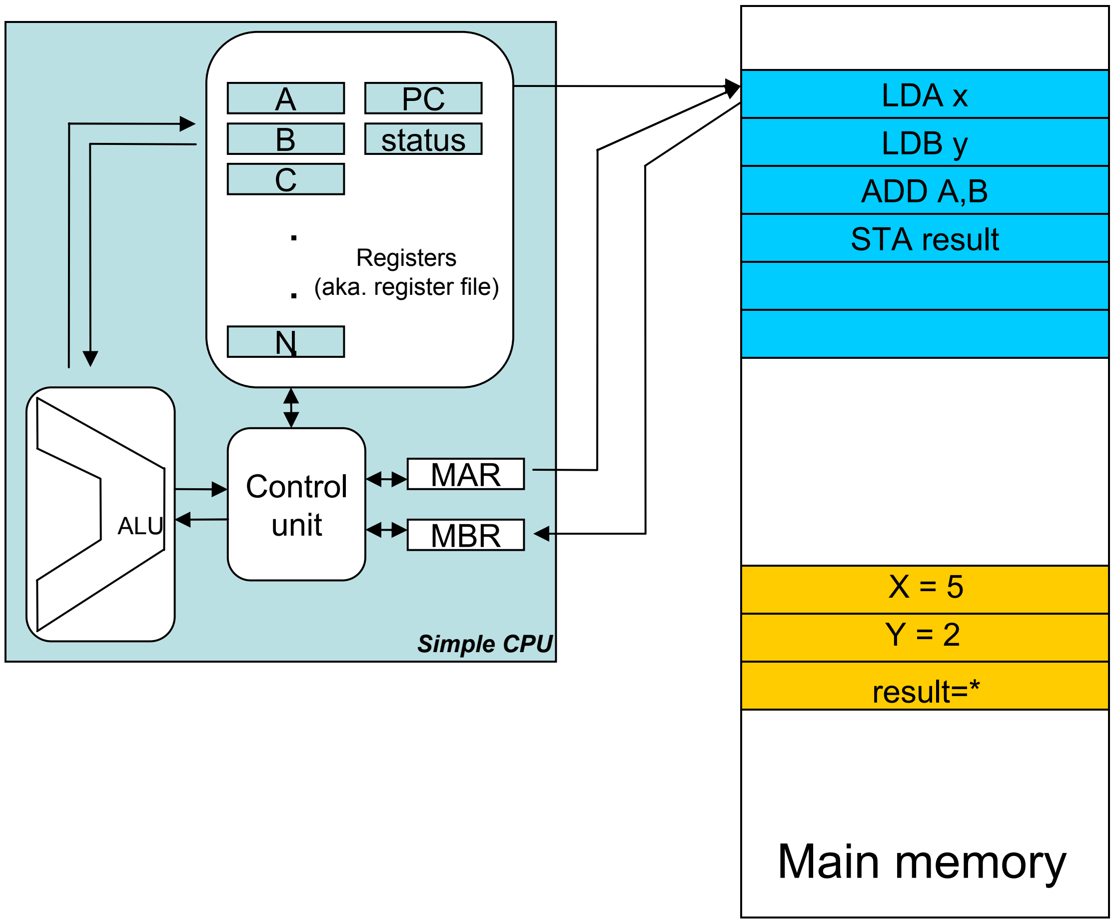
Fetch – Decode -Execute

* Sometimes a 4th stage is added

Fetch – Decode –Execute - Write

The Von Neumann Model

The fetch-decode-execute cycle



Fetch

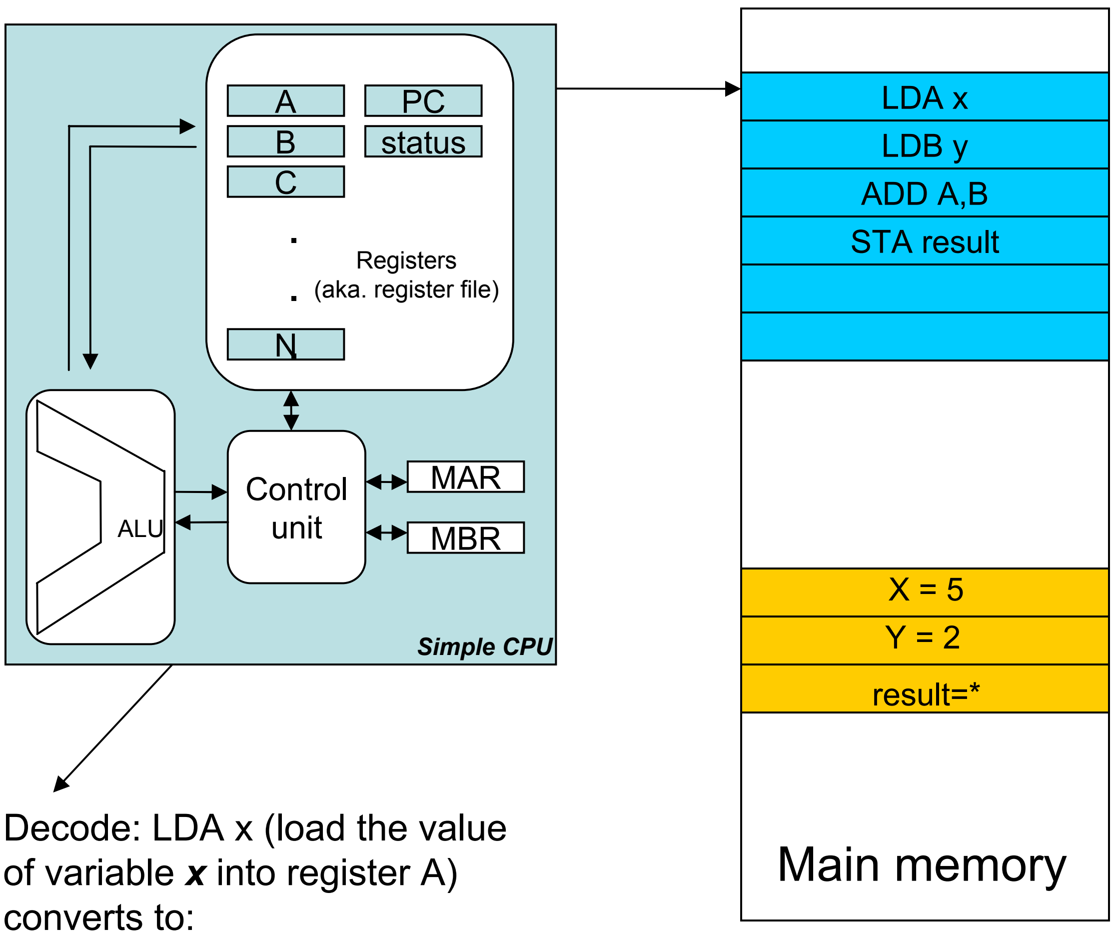
The contents of the program counter ***PC*** are loaded onto the ***MAR***

This selects the address of the next instruction (LDA x)

The instruction is read Into the ***MBR***

The Von Neumann Model

The fetch-decode-execute cycle



Decode: LDA x (load the value of variable ***x*** into register A) converts to:
- Move address of x to MAR
- Read x into MBR
- Store MBR into register A

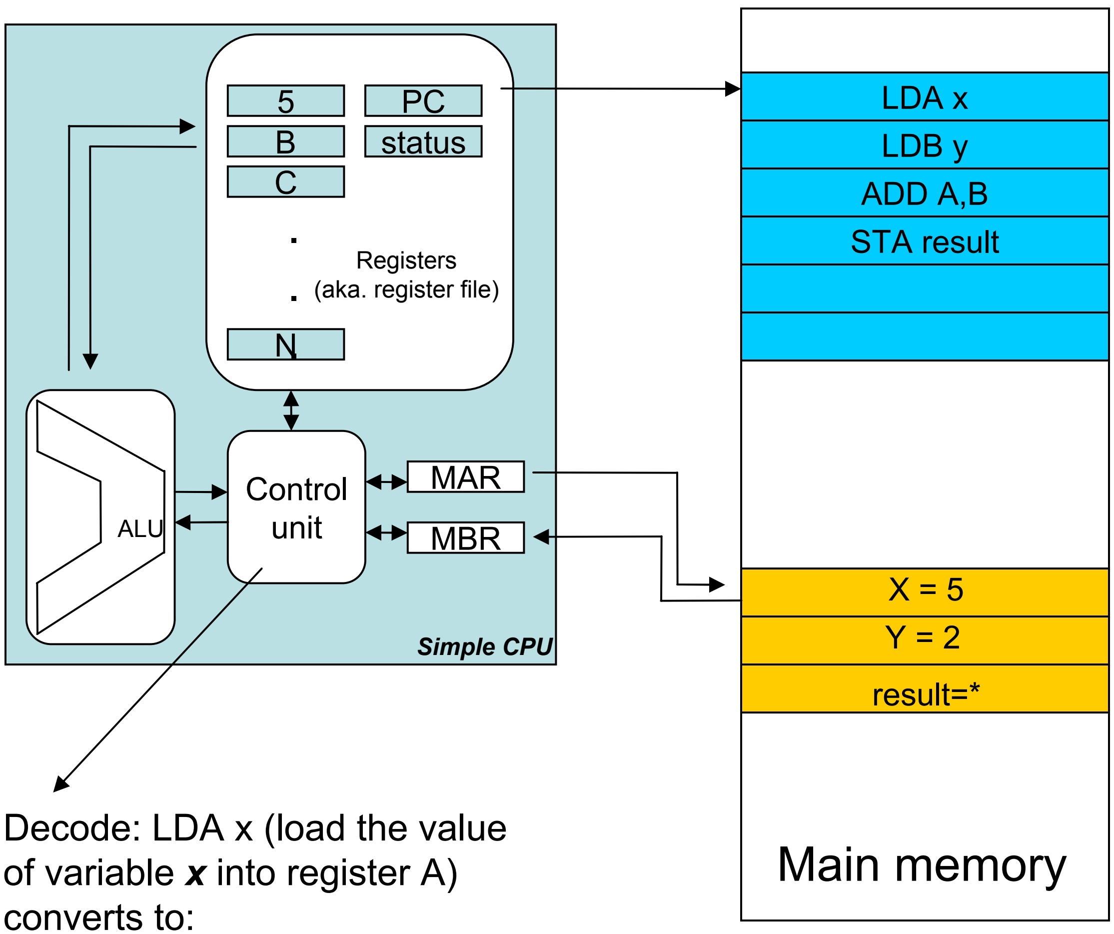
Decode

The instruction is converted Into a sequence of simple register/ALU operations

This sequence of operations is known as microcode

The Von Neumann Model

The fetch-decode-execute cycle



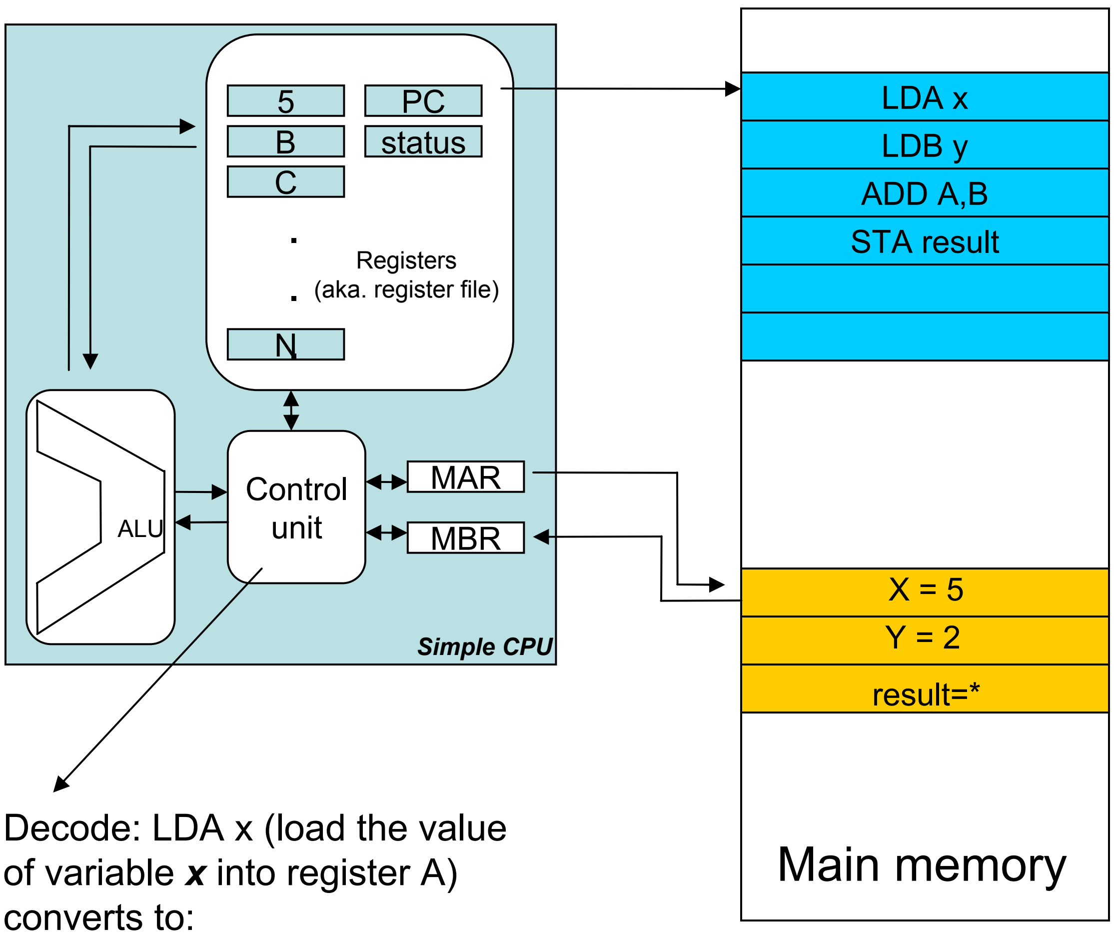
Decode: LDA x (load the value of variable ***x*** into register A) converts to:
- Move address of x to MAR
- Read x into MBR
- Store MBR into register A

Execute

The sequence of microcode instructions is executed

The Von Neumann Model

The fetch-decode-execute cycle



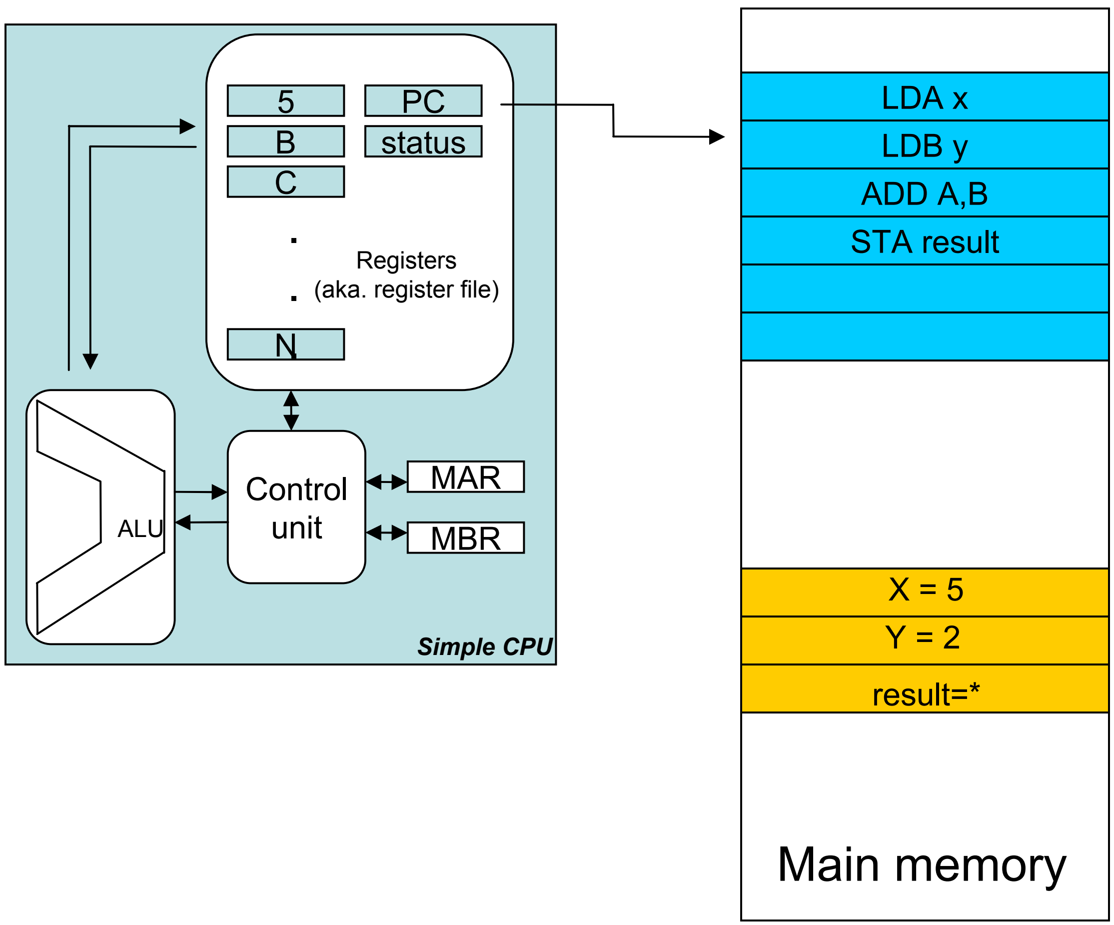
Decode: LDA x (load the value of variable ***x*** into register A) converts to:
- Move address of x to MAR
- Read x into MBR
- Store MBR into register A

Execute

The sequence of microcode instructions is executed

The Von Neumann Model

The fetch-decode-execute cycle

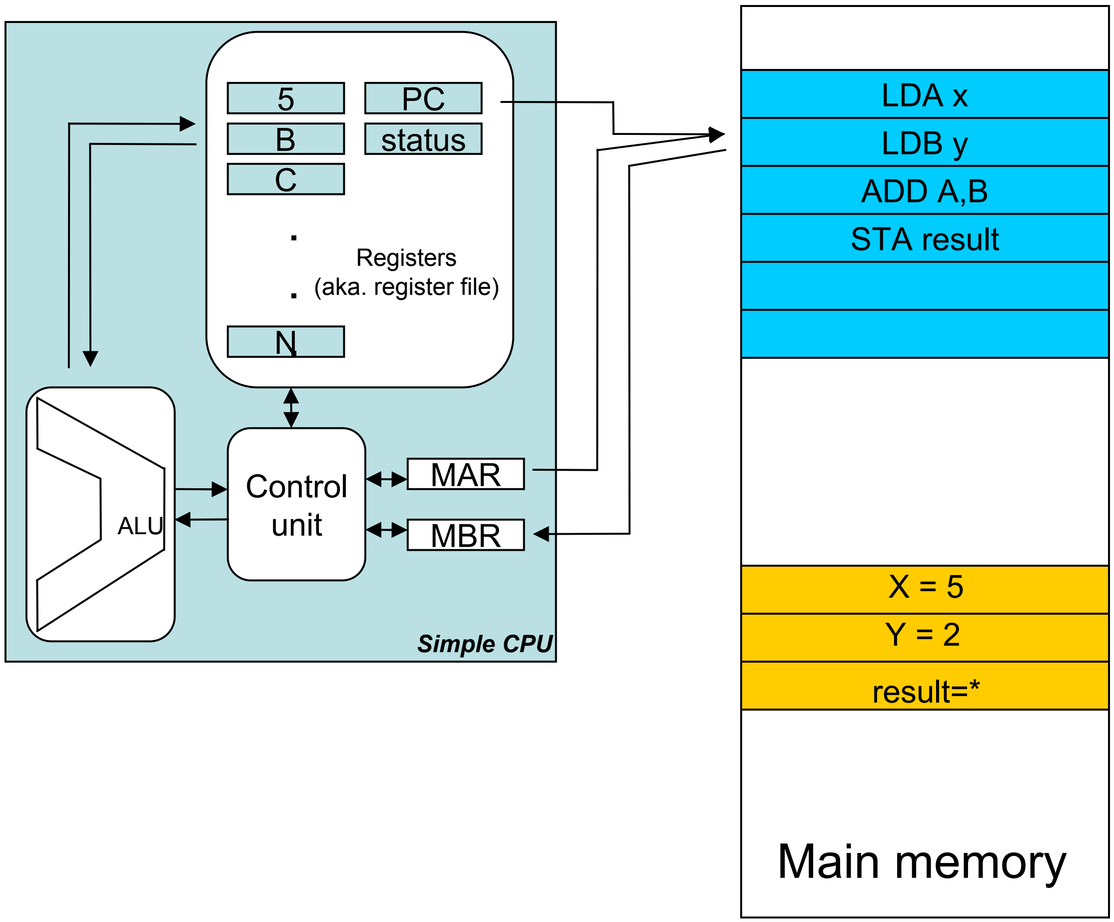


Execute

The program counter is incremented to point to the next instruction

The Von Neumann Model

The fetch-decode-execute cycle

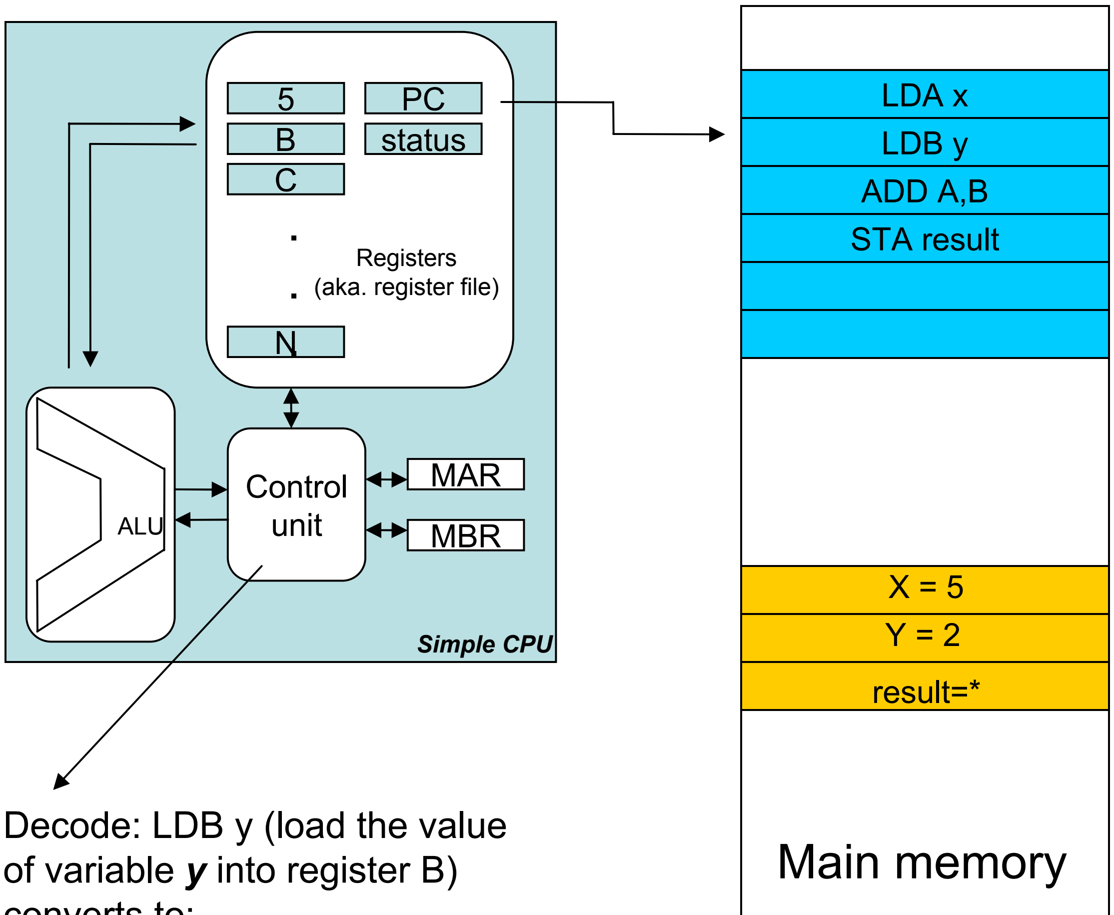


Repeat the cycle

Fetch LDB y

The Von Neumann Model

The fetch-decode-execute cycle



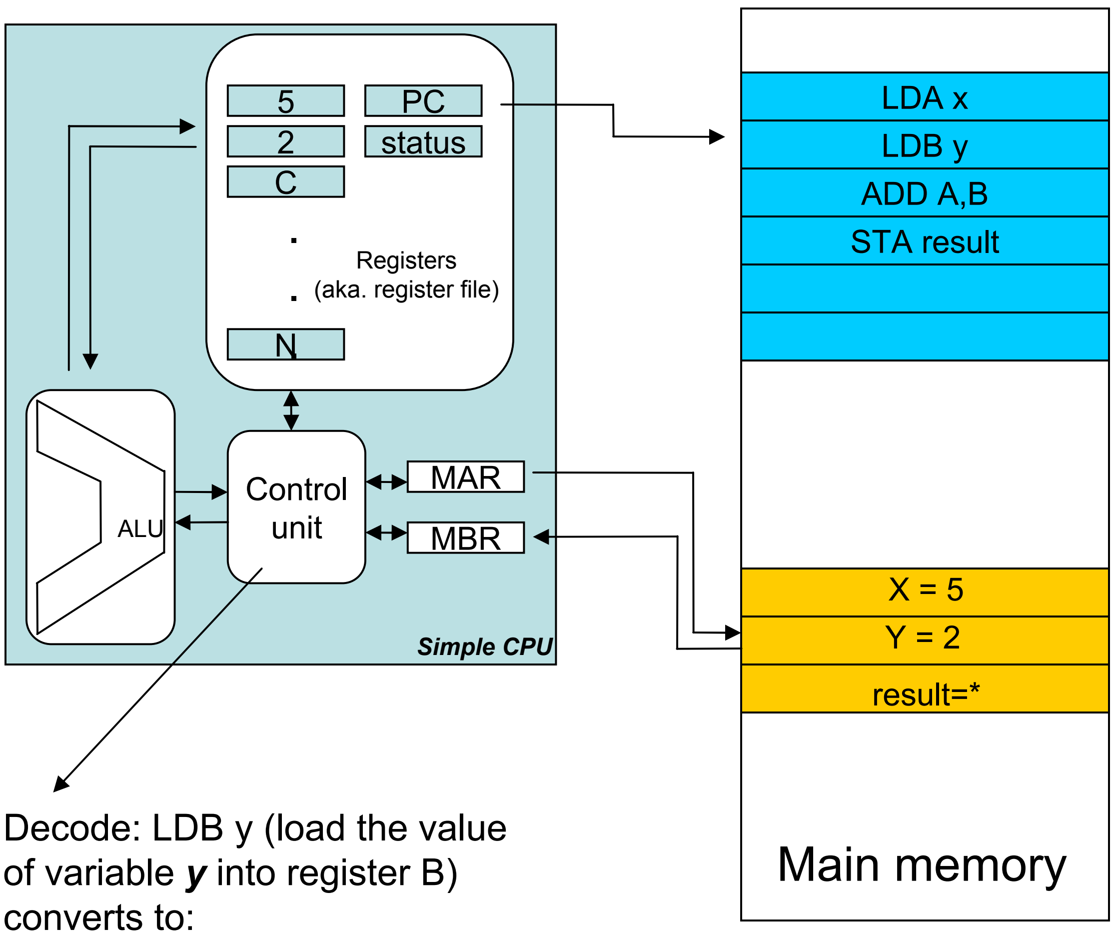
Repeat the cycle

Decode LDB y

Decode: LDB y (load the value of variable ***y*** into register B) converts to:
- Move address of y to MAR
- Read ***y*** into MBR
- Store MBR into register B

The Von Neumann Model

The fetch-decode-execute cycle



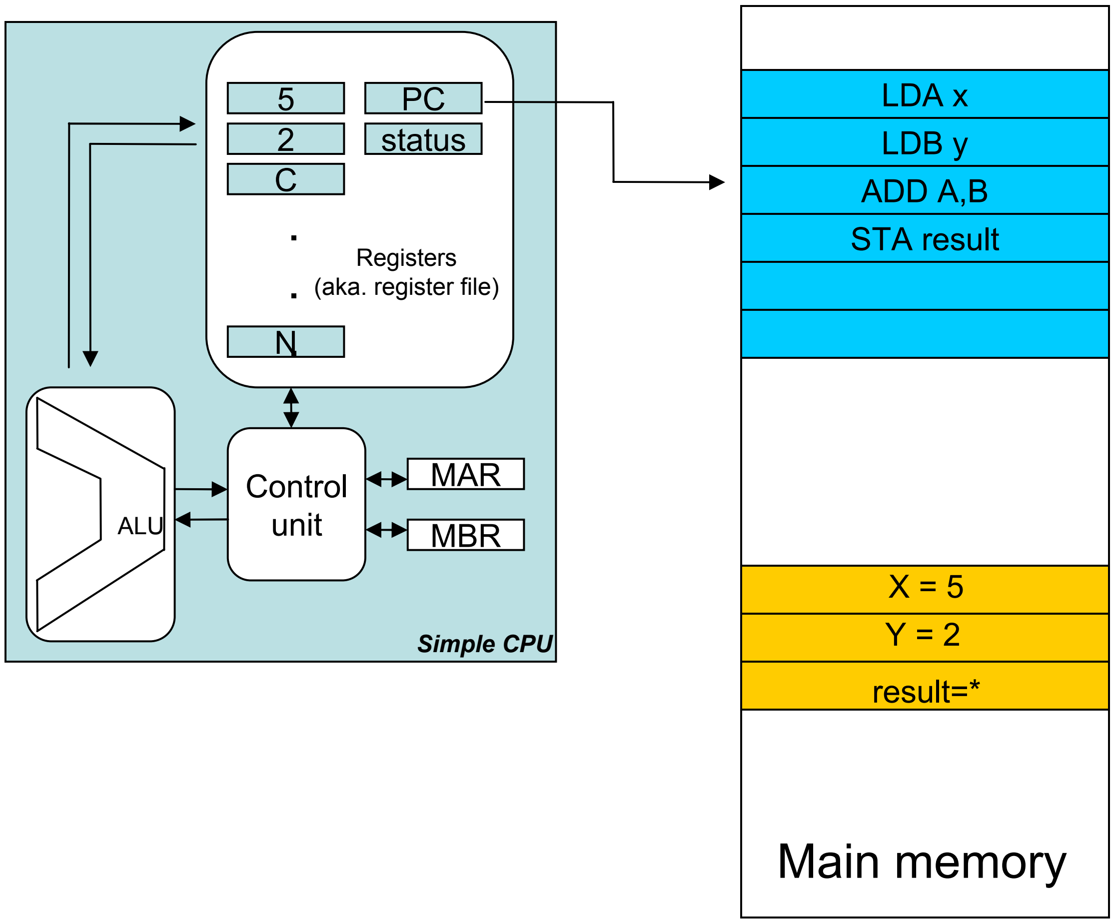
Decode: LDB y (load the value of variable ***y*** into register B) converts to:
- Move address of y to MAR
- Read ***y*** into MBR
- Store MBR into register B

Repeat the cycle

Execute microcode for LDB y

The Von Neumann Model

The fetch-decode-execute cycle

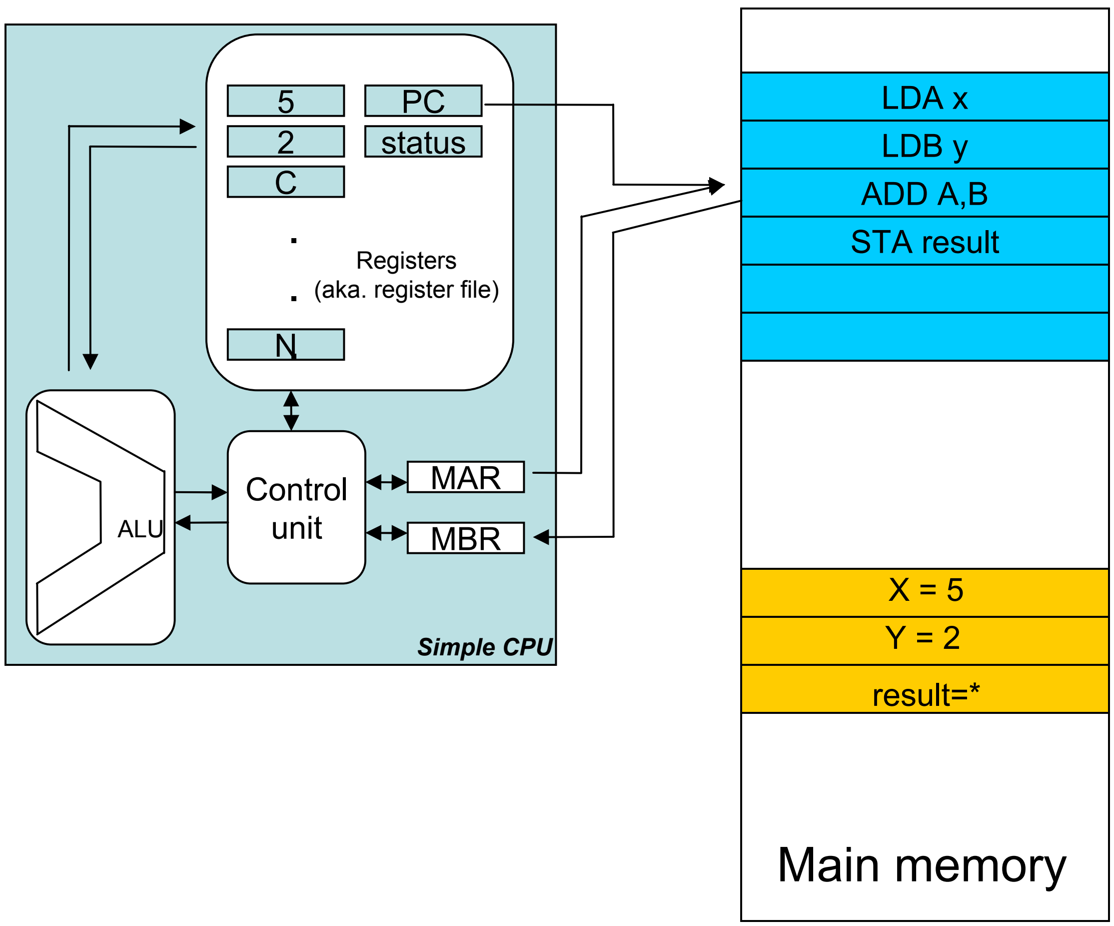


Repeat the cycle

Increment PC

The Von Neumann Model

The fetch-decode-execute cycle

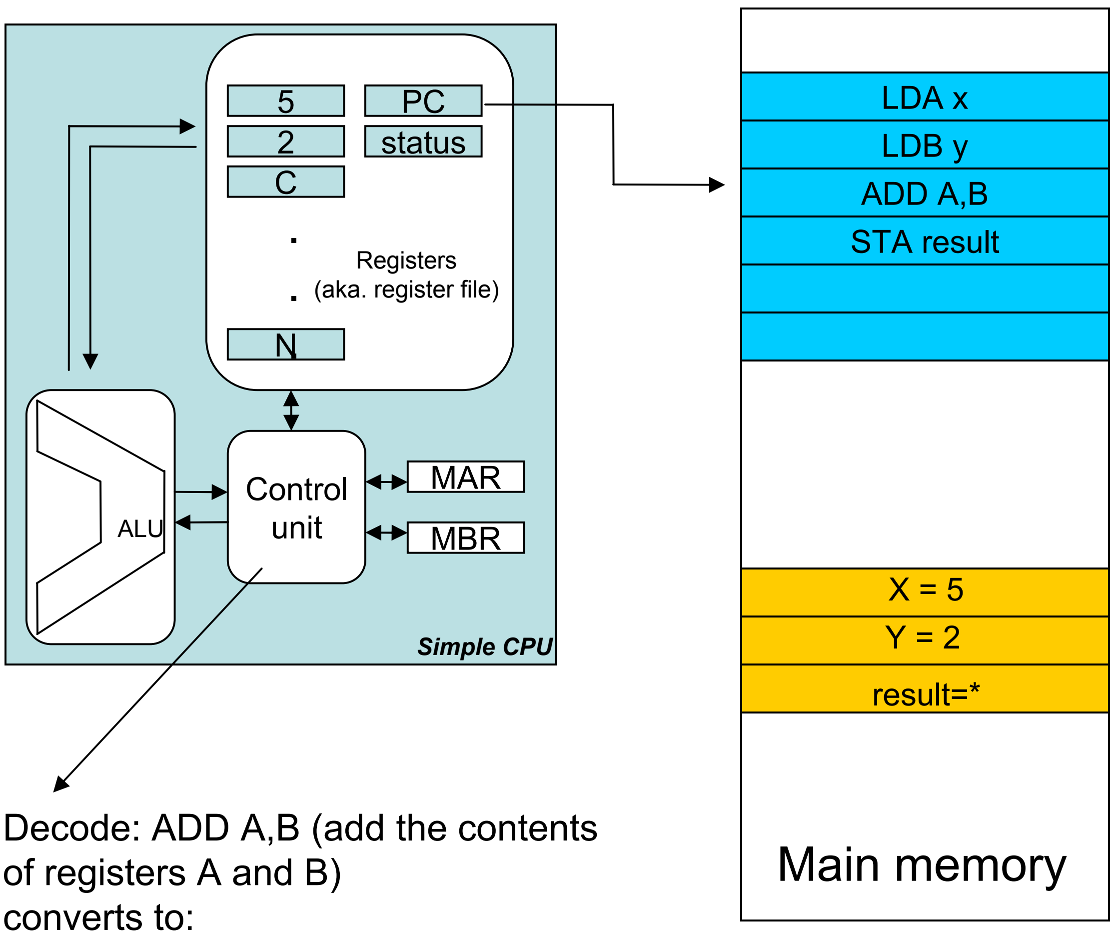


Repeat the cycle

Fetch ADD A,B

The Von Neumann Model

The fetch-decode-execute cycle



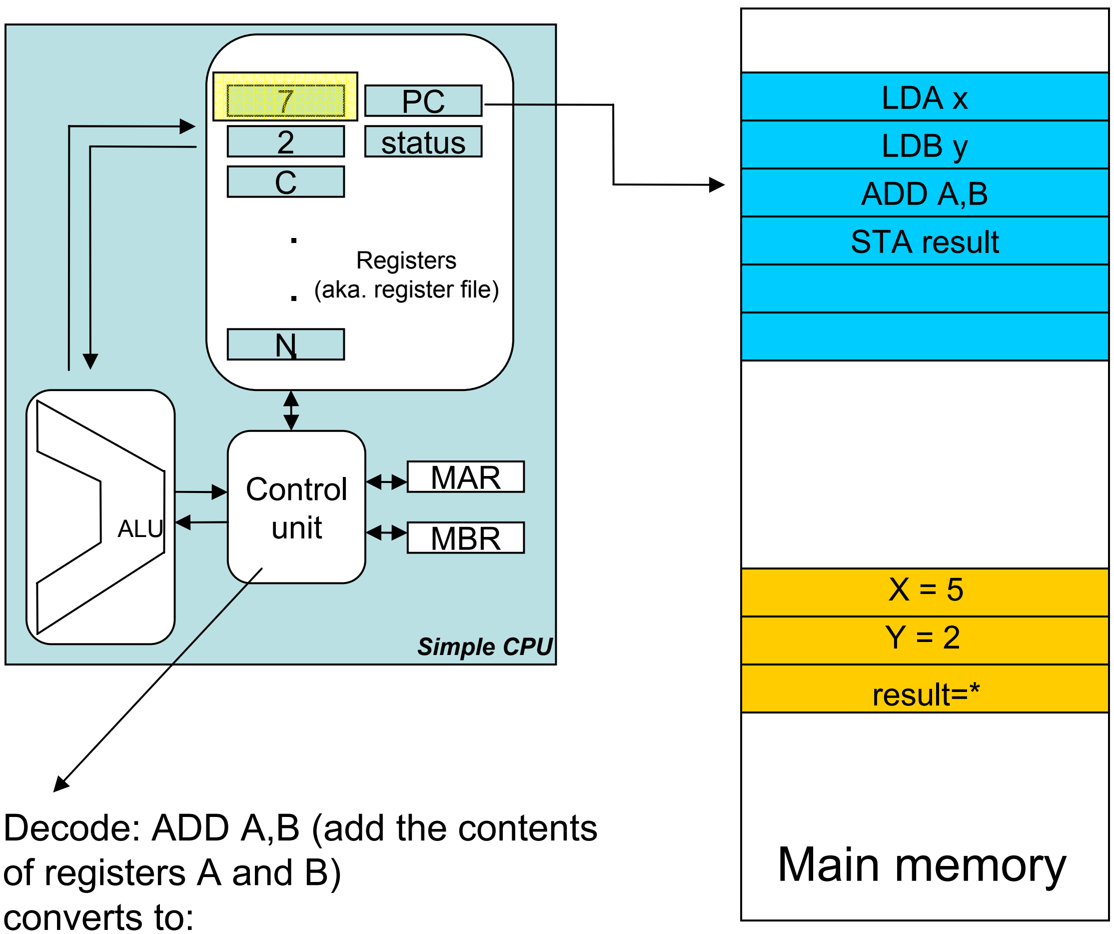
Decode: ADD A,B (add the contents of registers A and B)
converts to:
- ALU : Sum reg A, reg B
- Store result in A

Repeat the cycle

Decode ADD A,B

The Von Neumann Model

The fetch-decode-execute cycle



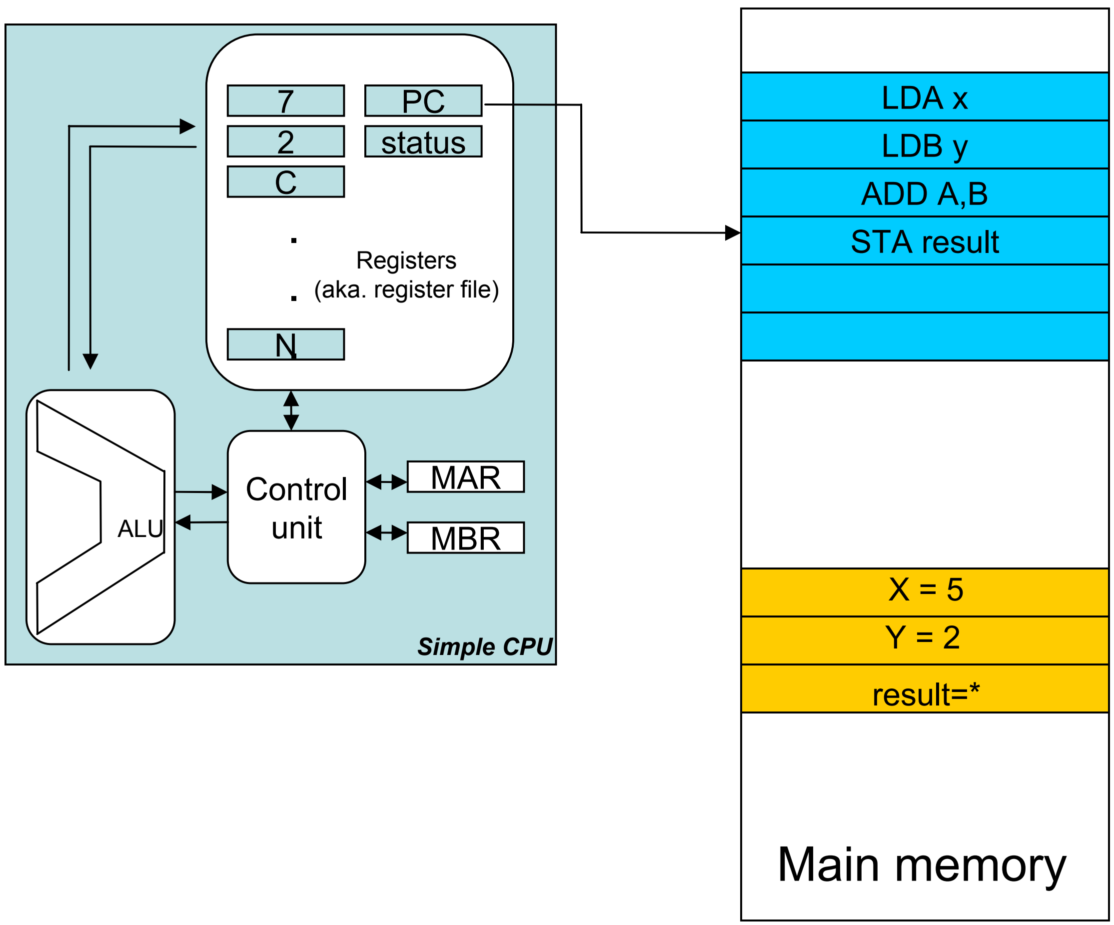
Decode: ADD A,B (add the contents of registers A and B)
converts to:
- ALU : Sum reg A, reg B
- Store result in A

Repeat the cycle

Execute microcode for ADD A,B

The Von Neumann Model

The fetch-decode-execute cycle

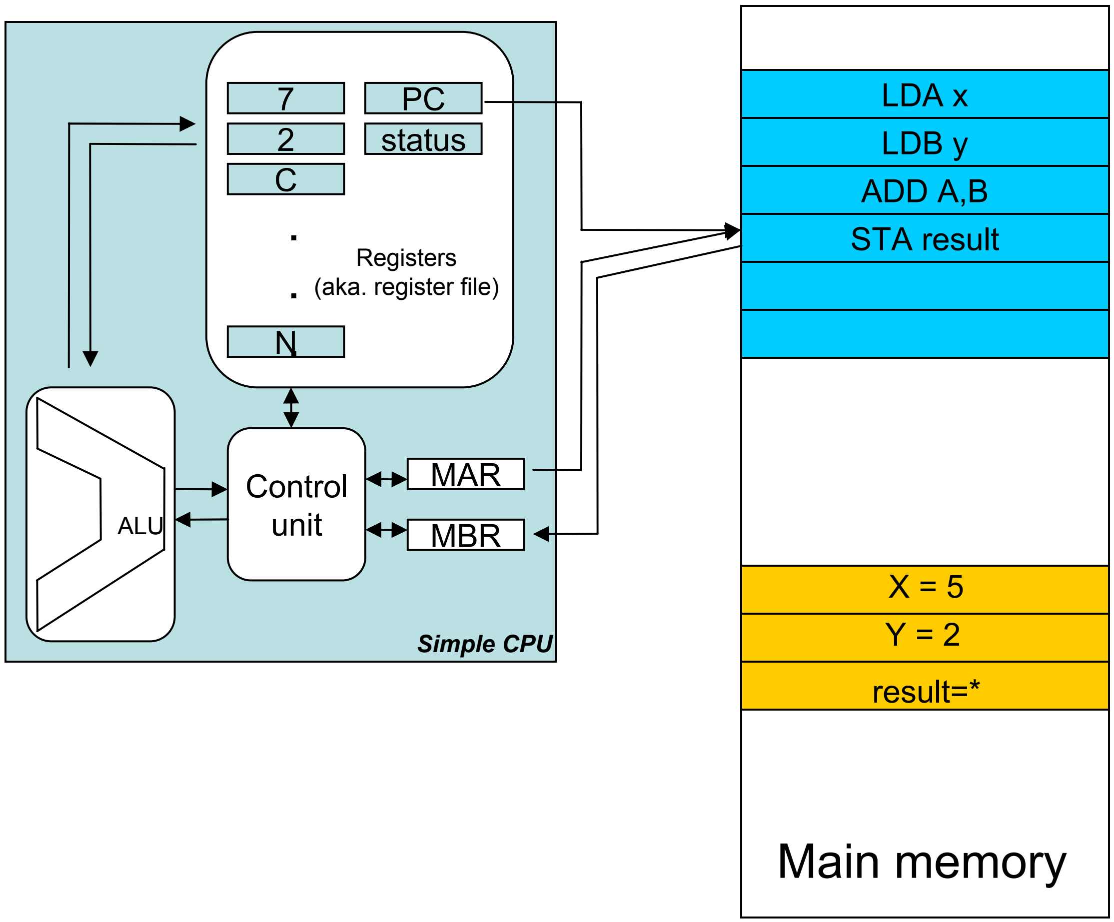


Repeat the cycle

Increment PC

The Von Neumann Model

The fetch-decode-execute cycle

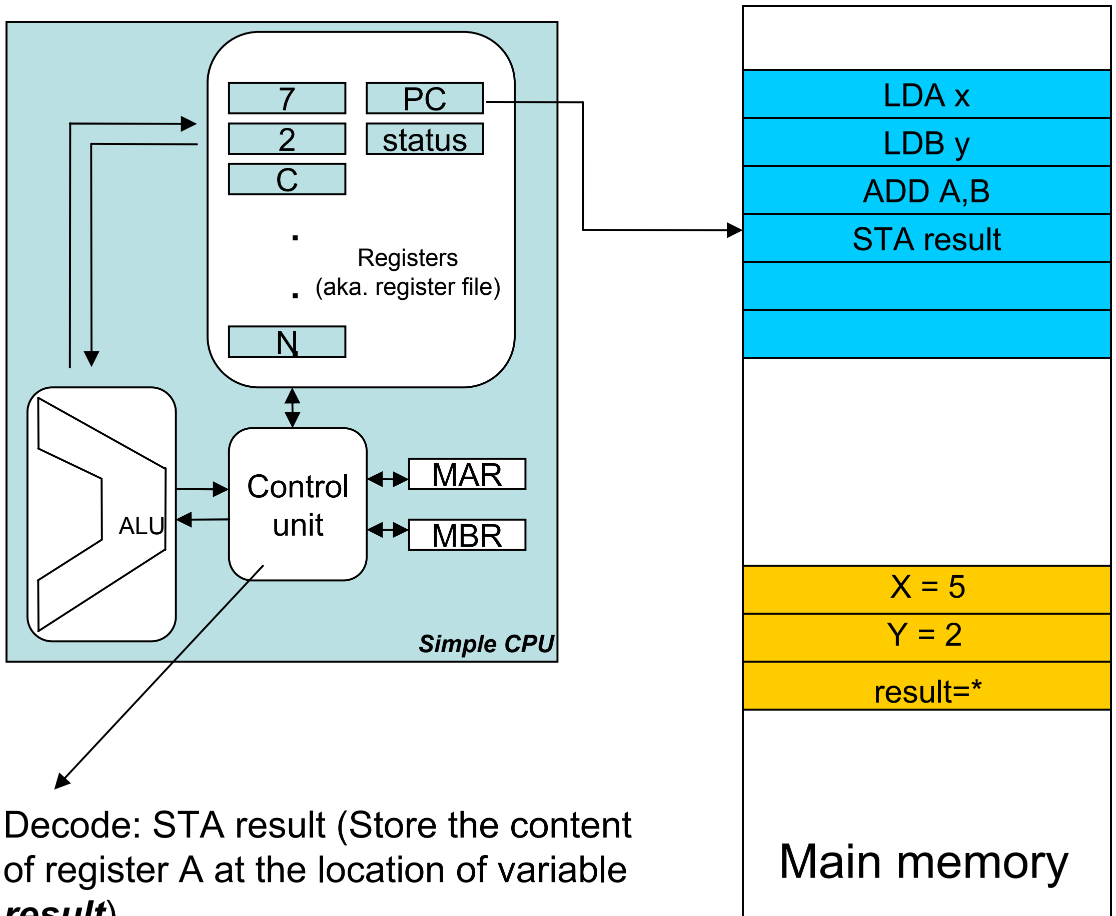


Repeat the cycle

Fetch STA result

The Von Neumann Model

The fetch-decode-execute cycle



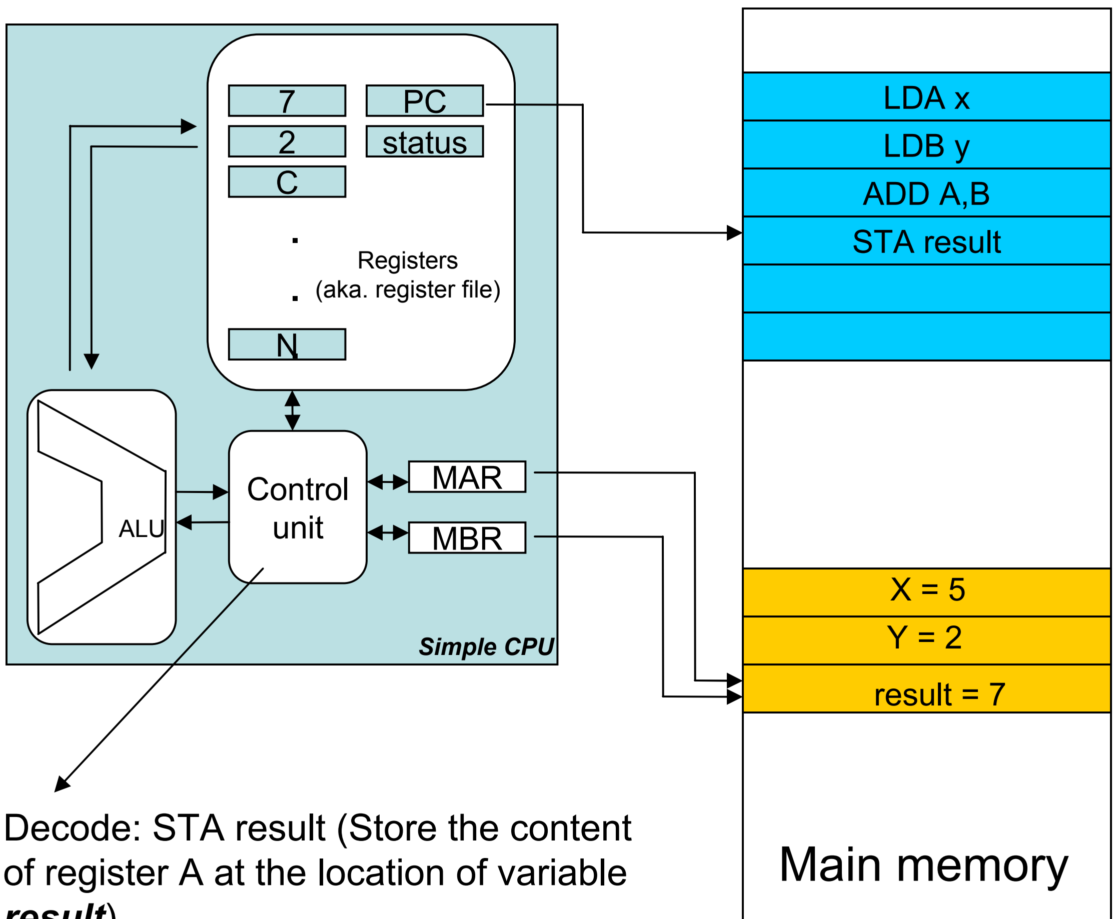
Repeat the cycle

Decode STA result

Decode: STA result (Store the content of register A at the location of variable ***result***)
converts to:
- Move address of result to MAR
- Store the contents of register A in the MBR
- Write MBR to memory

The Von Neumann Model

The fetch-decode-execute cycle



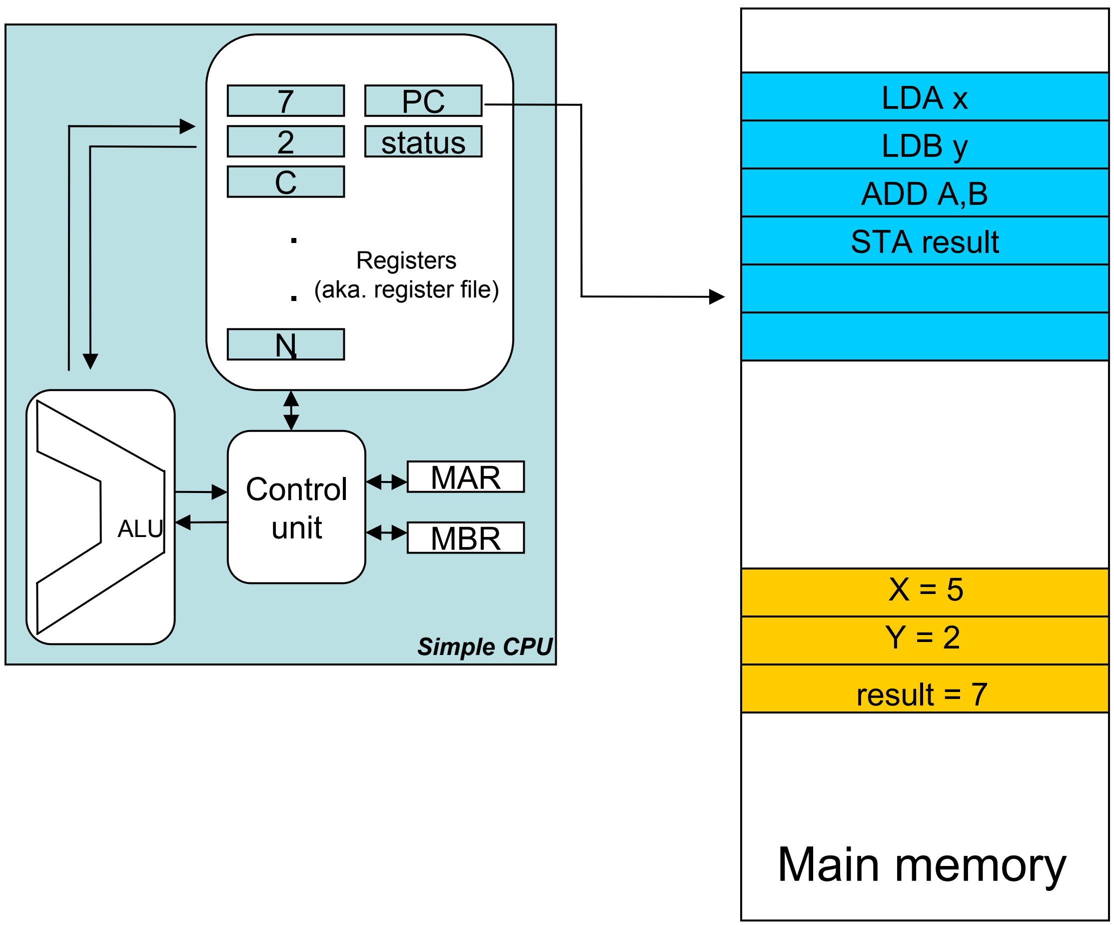
Decode: STA result (Store the content of register A at the location of variable ***result***)
converts to:
- Move address of result to MAR
- Store the contents of register A in the MBR
- Write MBR to memory

Repeat the cycle

Execute microcode for STA result

The Von Neumann Model

The fetch-decode-execute cycle



Repeat the cycle

Increment PC…

Fetch-Decode-Execute

This cycle is performed uninterruptedly by the CPU as long as the system is ***on***

*The computer is always doing *something* (i.e. unless sent to stand by, there is always some code being executed –what code?-)*

We need to know how to program the CPU (assembly language)

Having done that, we will know how to control every aspect of the computer's operation.

Non Von Neumann architectures

Are becoming more common

Parallel computation

* Parallel computers
* Neural Networks
* Genetic Algorithms
* Quantum computers