CSC 347

Computer Hardware and Maintenance

Lecture 3

Central Processing Unit (CPU)

Abhijit Saha, Ph.D.
Professor
Dept. of Computer Science and Engineering (CSE)
College of Engineering and Technology (CEAT)
IUBAT, Uttara, Dhaka
Email: asaha@iubat.edu

English is the First Language of IUBAT Campus

- This is mandatory for everyone including students in all interactions and communications as of January 1, 2014.
- If any student face difficulty, s/he is advised to contact Mr Nazmul Haque Khan (Room No: 224/A, Cell: 01727277166, Email: nazmul@iubat.edu, Ext: 460,) for arranging special spoken English training.
- Violation of English as the First Language in the Campus will lead to administrative and disciplinary action.
- All are urged to help each other to develop the Facility of Communicating in English as the First Language in the Campus

Introduction

- **Central Processing Unit (CPU)**
- The meaning of Central Processing Unit:
  - It is central, because it is the center of PC data processing
  - It is a processor, because it can work with data, do calculations, and move data
  - It is a unit, because it is a chip which contains billions of transistors
- CPU allows the processing of numeric data, meaning information entered in binary form, and the execution of instructions stored in memory
- CPU is frequently compared to the human brain

Introduction (Cont')

There are four functions of computer processor (CPU):

- CPU fetches and executes the commands
- It temporarily stores data being processed and intermediate results
- Controls all communication between RAM and all other input-output devices by interpreting data from and to the devices.
- It carries out all arithmetic calculations and performs logic operations

Introduction (Cont')

- First microprocessor (Intel 4004) was invented in 1971
  - It was a 4-bit calculation device with a speed of 108 kHz.
- In the early days of computing, a CPU only have a single core
- Multicore: Multiple processing (Dual-, Quad-, Hexa-, Octa-, Deca-, Dodeca-core..)
  - AMD's 64-core, with 128 threads, The **3rd Gen Ryzen ThreadRipper 3990X desktop PC processor** fastest CPU in 2021
  - **AMD's monster 64-core/128-thread Ryzen Threadripper Pro 5995WX**, but that's a High-End Desktop (HEDT) processor in 2022
  - **12th Gen Intel Core i9-12900KS (16-cores)** World's Fastest Desktop Processor

Introduction (Cont')

Table I. Processor Core Types with examples

Processor type	Number of core	Types of cores used	Examples of processors
Single-core	1	1 core	Intel Celeron (B720), AMD-Embedded G-Series T40R
Dual-core	2	1 Dual	Intel Core Duo, Intel Pentium Dual Core, AMD X2
Quad-core	4	1 Quad	PowerPC G5, Intel Core 2 Quad, Intel Nehalem, AMD Phenom X4
Hexa-core	6	1Qaud, 1Dual	Intel Core i5-8650, AMD Ryzen 5 PRO 1600
Octa-core	8	2Qaud	AMD Ryzen 7 4980U, Intel Core i7-9800X, Qualcomm Snapdragon 665
Deca-core	10	2Qaud, 1Dual	Intel Core i9-9820X, MediaTek Helio X30 (MT6799), Cavium CN5745-1000 SSP

Introduction (Cont')

- The CPU is a microprocessor that is an integrated circuit made up with billions of transistors
- Not all the microprocessors are CPUs
  - There are:
    - Numeric (floating point) Processing Unit (NPUs),
    - Graphical Processing Unit (GPUs) and
    - Accelerated Processing Unit (APUs)

CPU Components and Function

- There are seven components that ensure the processor work effectively and efficiently

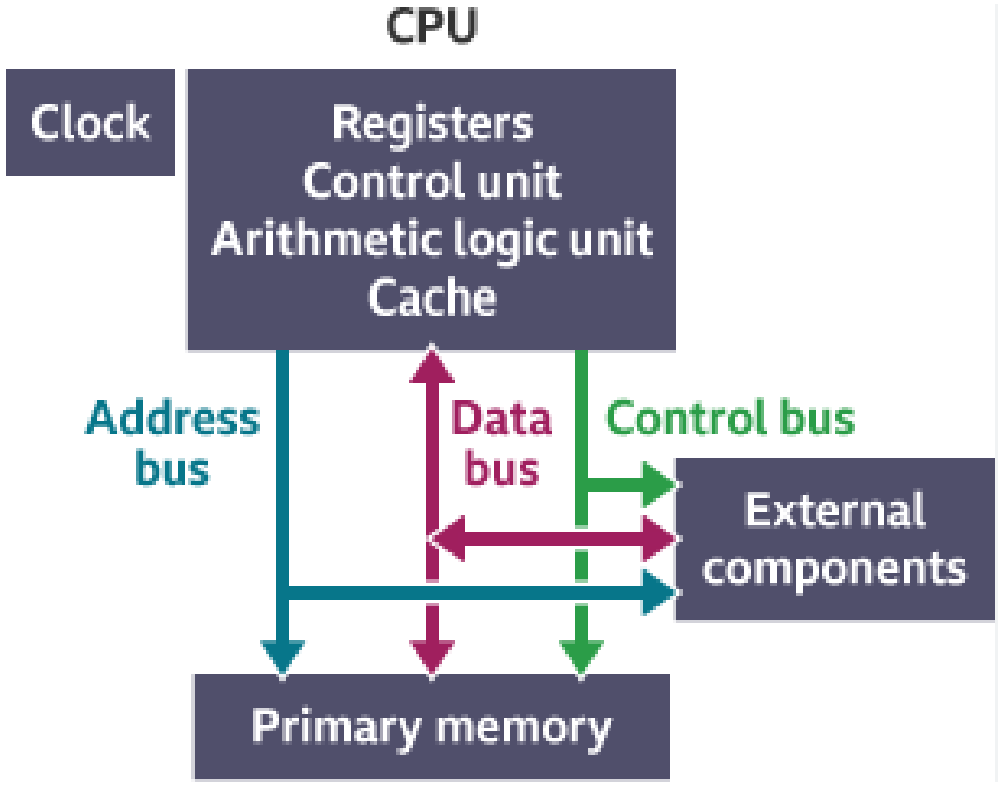


Figure 3.1 Common CPU Components

CPU Components and Function (Cont')

1. Control Unit:

- It controls all activities/operations of the different units but does not carry out any actual data processing operation
- Control unit transfers data or instruction among different units of a computer system
- It receives the instructions from the memory, interprets them and sends the operation to various units as instructed
- It is also responsible for communicating with all input and output devices for transferring or receiving the instruction from the storage units
- Therefore, the control unit is the main coordinator since it sends signals and find the sequence of instructions to be executed

CPU Components and Function (Cont')

2. Arithmetic Logic Unit:

- ALU is where all calculation and logic operations happen
- It is made up of arithmetic and logic units:
  - The arithmetic unit deals with basic arithmetic operations such as addition, subtraction, multiplication, and division
  - The logic unit caters to logic operations such as AND, OR, NOT, and NOR. Therefore, ALU is the major part of the computer system which handles different calculations
  - Depending on the design of ALU it makes the CPU more powerful and efficient.

CPU Components and Function (Cont')

3. Main Memory (RAM) :

- RAM is counted as part of the CPU but it is not integrated within the microchip that carries the other component
- It is part since the CPU cannot process data without having somewhere to store the data
- Main memory (RAM) is used temporally to store data that is required for processing within the CPU

CPU Components and Function (Cont')

4. CPU Registers:

- When the processor executes instructions, data is temporarily stored in small, local memory locations of 8, 16, 32 or 64 bits called registers
- Depending on the type of processor, the overall number of registers can vary from about ten to many hundreds
  - **Program counter (PC):** stores the next instruction
  - **Memory Address Register (MAR):** stores the address of the next instruction
  - **Memory Data Register (MDR):** contains data that is to be read/written to or from the address
  - **Current Instruction Register (IR):** stores instructions being executed currently
  - **Accumulator Register (AR):** used to store intermediate results that are produced

CPU Components and Function (Cont')

5. Computer Buses:

- These are pathways that data travels when the CPU is communicating with RAM and other input-output devices.
- There are 3 types of Computer Buses.

Address BUS: it carries the address of where data is going to and coming from in the RAM

Control BUS: it carries control signal data that is used to control the communication between devices

Data BUS: it carries the real data that is being transmitted from one location to another

CPU Components and Function (Cont')

6. Cache Memory:

- Cache memory also known as CPU memory is a high-speed intelligent memory buffer that temporarily stores data the processor needs
- This allows the computer processor to execute instructions faster
- Cache reduces retrieval time compared to if data was accessed directly from the main memory (RAM)
- It is said to be intelligent since it not only stores data but it stores the data that will be used by the processor on its next execution
- It is made up of Static RAM which is either integrated into the processor or in a separate chip within the computer motherboard
- Its main function is to increase the processor speed
- There are 3 levels of cache memory L1, L2, and L3. In their architecture, they are arranged in such a way that the processor looks for data from cache L1 up to L3 in that order

CPU Components and Function (Cont')

7. CPU Clock:

- This is an electronic pulse that determines the number of cycles that a CPU executes instructions per second
- The cycles are measured in Hertz. The current computer processors have a speed of Gigahertz (Billion cycles per second)
- The clock speed is given together with processor types and generation
- A system clock or timer uses a quartz crystal oscillator to generate pulses
- These pulses are counted as the number of cycles per second (cycles are counted into Hertz)
- A modern computer has a million (MHz)or billion hertz per second(GHz). Example: A 3.2GHz processor means it has 3.2 billion cycles per second

Operations

• Nearly, all CPUs follow the fetch, decode and execute steps in their operation, which are collectively known as the instruction cycle.

FETCH :

• Fetch involves retrieving an instruction from program memory

• The instruction's location (address) in program memory is determined by a program counter (PC), which stores a number that identifies the address of the next instruction to be fetched

• Retrieved from relatively slow memory, causing the CPU to stall while waiting for the instruction to be returned

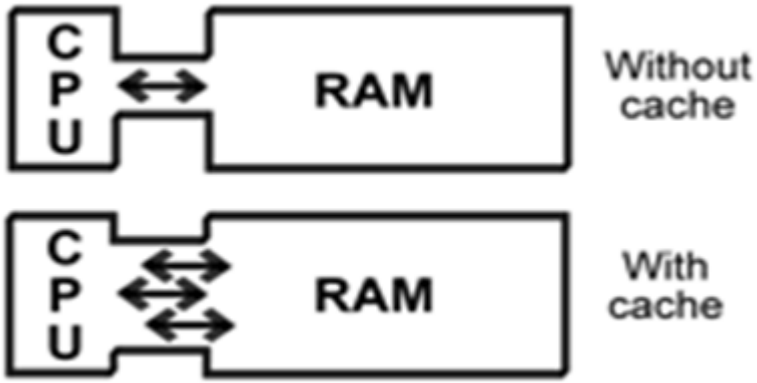
• Solution: **Cache Memory** and **Pipeline architectures** are utilized

Cache Memory

- Computer's main memory is slower than that of the processor (Speed Conflict)
- CPU cache used by the central processing unit of a computer to reduce the average time to access memory
- Cache is a smaller, faster memory that stores copies of the most frequently used data.

Cache RAM is much faster than normal RAM

- **Bottleneck Problem**



Cache hit

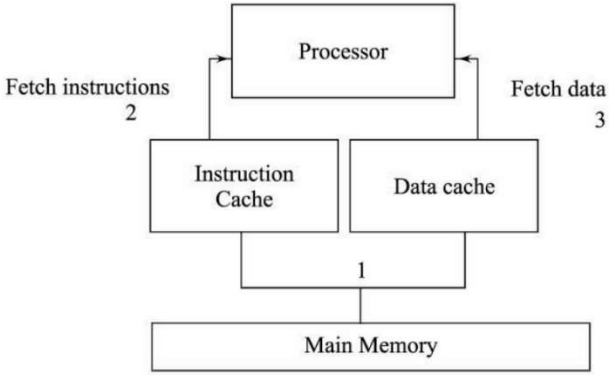
Cache miss

A cache increases the CPU's capacity to fetch the right data from RAM

Cache Memory (Cont')

L1 Cache

- Level 1 or L1 cache resides in each core
- Level 1 cache is typically 8, 16, 20, 32, 64,128 or 256 Kbytes, which operates at the same clock frequency as the rest of the CPU
- However, for caches, data and instructions are often stored in separate data and instruction caches

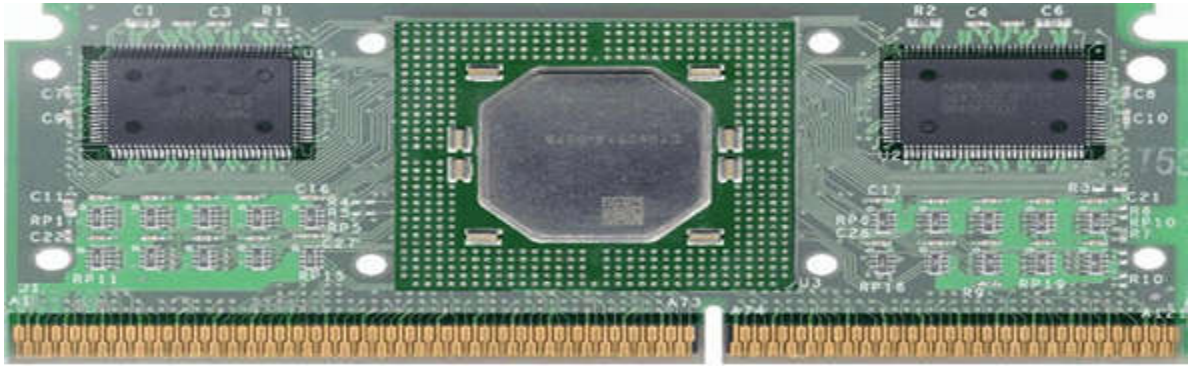


  - **Instruction cache** — stores the executable instruction which makes the instruction fetch operation faster
  - **Data cache** — stores the data to process which speeds up the data fetch operation
- When a cache contains both instructions and data, it is called a unified cache

Cache Memory (Cont')

L2 Cache

- L2 cache is larger and slower than the L1 cache
- It resides in the core or in the motherboard.
- The level 2 cache is normally much bigger such as 256, 512 or 1024 KB
- In the earlier processor generations, the L2 cache was placed *outside* the chip: either on the motherboard (as in the original Pentium processors), or on a special module together with the CPU (as in the first Pentium II's)



An old Pentium II module. The CPU is mounted on a rectangular printed circuit board, together with the L2 cache, which is two chips here. The whole module is installed in a socket on the motherboard. But this design is no longer used.

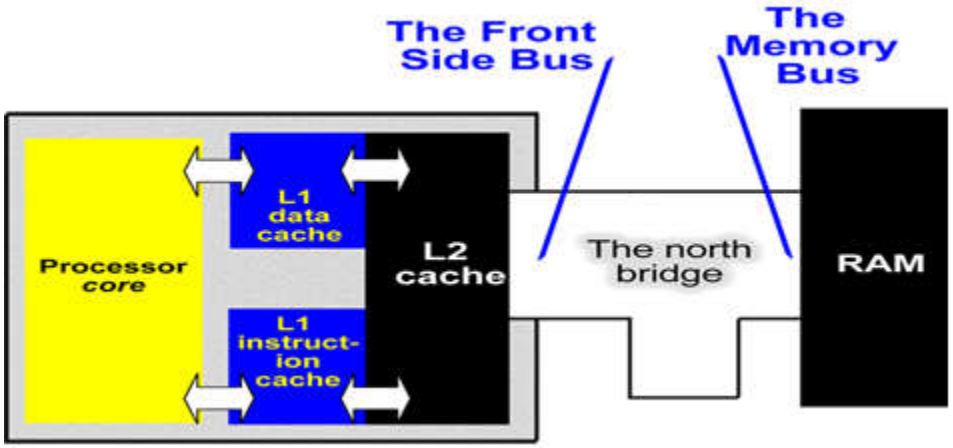
- Latest Intel Core processors have L2 cache integrated into core along with L1 cache

Cache Memory (Cont')

Uses of L1 and L2 Caches

Table I: Level 2 cache has integrated into the actual CPU. Traditionally, L2 cache is connected to the front side bus that connects it to the chipset's north bridge and RAM:

CPU	L2 cache
Pentium, K5, K6	External, on the motherboard
Pentium Pro	Internal, in the CPU
Pentium II, Athlon	External, in a module close to the CPU
Celeron (1st generation)	None
Celeron (later gen.), Pentium III, Athlon XP, Duron, Pentium 4	Internal, in the CPU



The way CPU uses L1 cache and L2 cache has crucial significance for its utilization of the high clock frequencies

Cache Memory (Cont')

L1 and L2 Caches (Powerful Bus)

- The bus between the L1 and L2 cache is presumably the place in the processor architecture which has the greatest need for high bandwidth

Table II: Theoretical calculations of the bandwidth between the L1 and L2 cache

CPU	Bus width	Clock frequency	Theoretical bandwidth
Intel Pentium III	64 bits	1400 MHz	11.2 GB/s
AMD Athlon XP+	64 bits	2167 MHz	17.3 GB/s
AMD Athlon 64	64 bits	2200 MHz	17.6 GB/s
AMD Athlon 64 FX	128 bits	2200 MHz	35.2 GB/s
Intel Pentium 4	256 bits	3200 MHz	102 GB/s

Theoretical bandwidth = Bus width * Clock frequency / 8

Cache Memory (Cont')

L1 and L2 Caches (Different systems)

- AMD have settled on a fairly large L1 cache of 128 KB, while Intel continue to use relatively small (but efficient) L1 caches.
- On the other hand, Intel uses a 256 bit wide bus on the "inside edge" of the L2 cache in the Pentium 4, while AMD only has a 64-bit bus (see Table II).
- AMD uses *exclusive caches* in all their CPU's. It means that the same data can't be present in both caches at the same time, and that is a clear advantage. It's not like that at Intel.
- However, the Pentium 4 has a more advanced cache design with *Execution Trace Cache* making up 12 KB of the 20 KB Level 1 cache. This instruction cache works with *coded instructions*.

Table III. The most common processors and their caches

CPU	L1 cache	L2 cache
Athlon XP	128 KB	256 KB
Athlon XP+	128 KB	512 KB
Pentium 4 (I)	20 KB	256 KB
Pentium 4 (II, "Northwood")	20 KB	512 KB
Athlon 64	128 KB	512 KB
Athlon 64 FX	128 KB	1024 KB
Pentium 4 (III, "Prescott")	28 KB	1024 KB

Cache Memory (Cont')

L1 and L2 Caches (Latency)

- All RAM storage has a certain latency, which means that a certain number of clock ticks (*cycles*) must pass between, for example, two reads. L1 cache has less latency than L2 resulting L1 cache more efficient

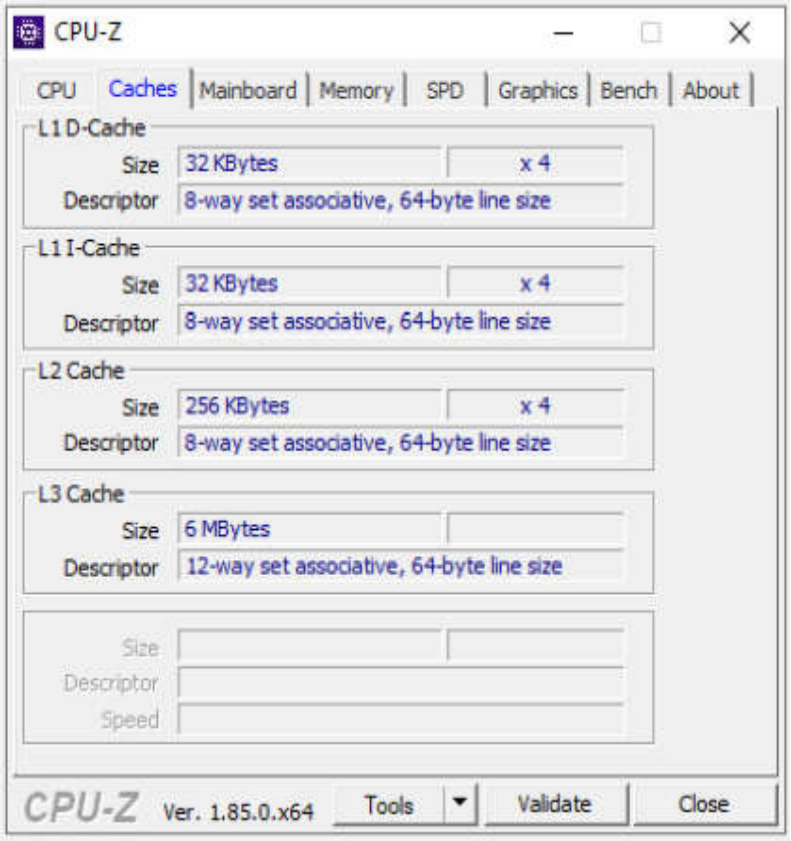
Table IV. Latency leads to wasted clock ticks; the fewer there are of these, the faster the processor will appear to be

Latency	Pentium II	Athlon XP	Pentium 4
L1 cache:	3 cycles	3 cycles	2 cycles
L2 cache:	18 cycles	6 cycles	5 cycles

Cache Memory (Cont')

L3 Caches

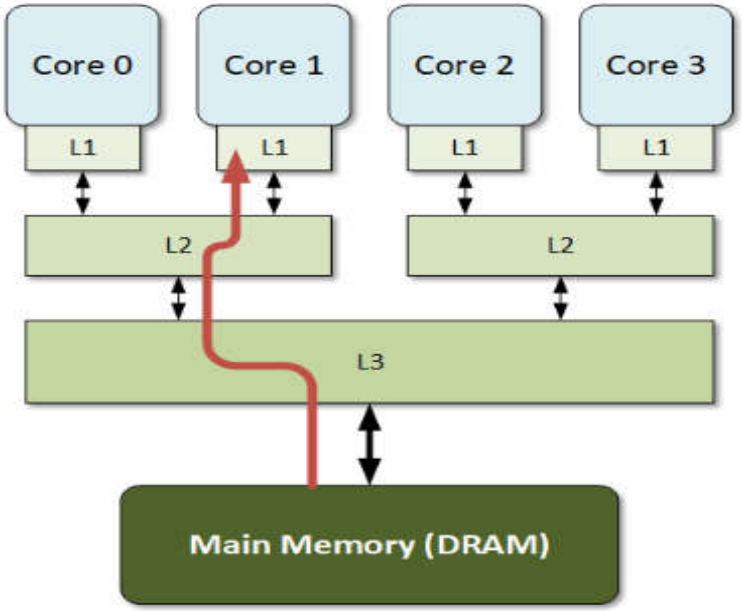
- In **the early days**, the L3 memory cache was actually found on the motherboard.
- This was a very long time ago, back when most CPUs were just single-core processors.
- Now, the L3 cache in your CPU can be massive, with top-end consumer CPUs featuring L3 caches up to 32MB.
- Some server CPU L3 caches can exceed this, featuring up to 64MB.

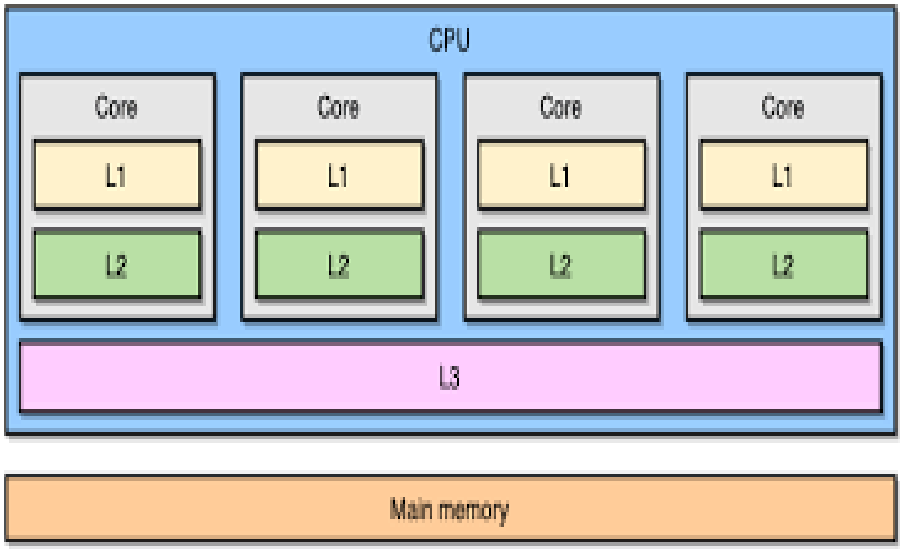


CPU memory cache levels for an Intel Core i5-3570K CPU

Cache Memory (Cont')

Multiple Levels of Cache





Note: Shared L2 Cache (left Fig.) between cores, and
L2 cache dedicated to each core (right Fig.)

Figure Cache in multi-core CPU

Pipelines

- Pipelining is technology that improves instruction execution speed by putting the steps into parallel
- Instructions are sent from the software and are broken down into micro-ops (smaller sub-operations) in the CPU
  - This decomposition and execution takes place in a pipeline
- In the Pipeline:
  - First, CISC instructions are decoded and converted into more digestible micro instructions
  - then, these are processed at a time in the rest of the pipeline
- The pipeline is made up of a number *stages*
- Older processors have only a few stages, while the newer ones have many
- At each stage "something" is done with the instruction, and each stage requires one clock tick from the processor

Pipelines (Cont')

- **Pipeline Stages** RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set. Following are the 5 stages of the RISC pipeline with their respective operations:

Stage 1 (Instruction Fetch) In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.

Stage 2 (Instruction Decode) In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.

Stage 3 (Instruction Execute) In this stage, ALU operations are performed.

Stage 4 (Memory Access) In this stage, memory operands are read and written from/to the memory that is present in the instruction.

Stage 5 (Write Back) In this stage, computed/fetched value is written back to the register present in the instructions.

Pipelines (Cont')

- **Basic five-stage pipeline in a RISC machine**
  - IF = Instruction Fetch,
  - ID = Instruction Decode,
  - EX = Instruction Execute,
  - MEM = Memory Access,
  - WB = Write Back

The vertical axis is successive instructions; the horizontal axis is time. So in the green column, the earliest instruction is in WB stage, and the latest instruction is undergoing instruction fetch.

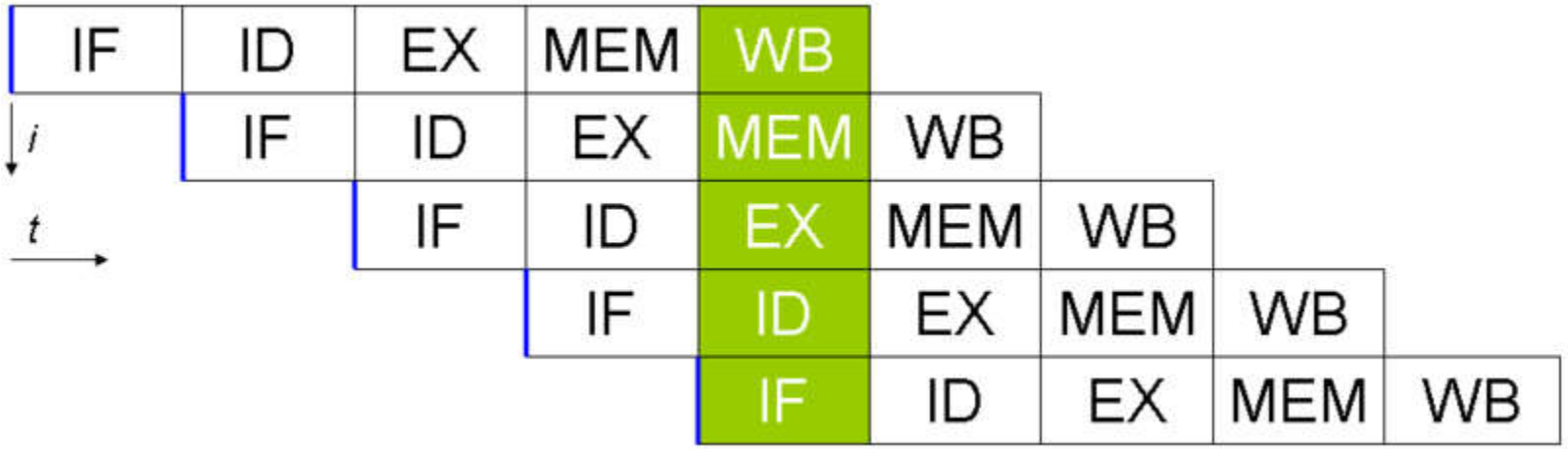


Figure 10 Stages in the pipeline

Pipelines (Cont')

- Pentium 4 and AthlonXP can decode about 2.5 instructions per clock tick
- First Pentium 4 has several long pipelines, allowing the processor to hold up to 126 instructions in total
  - These all being processed at the same time, but at different stages of execution

Table V. By making use of more, and longer pipelines, processors can execute more instructions at the same time

CPU	Instructions executed at the same time
AMD K6-II	24
Intel Pentium III	40
AMD Athlon	72
Intel Pentium 4	126

Pipelines (Cont')
What is actually happens in the pipeline?

- Two types of unit:
  - ALU (Arithmetic and Logic Unit)
  - FPU (Floating Point Unit)
- ALU is the calculating device that does operations on whole numbers (*integers*)
- When it comes to numbers with many decimal places, the ALU chokes, and can take a very long time to process the operations
- FPU is used to relieve the load
- An FPU is a number cruncher, specially designed for floating point operations
- Typically, several ALU's and FPU's are there in same processor
- CPU also has other operation units, e.g. LSU (Load/Store Unit)

Pipelines (Cont')

What is actually happens in the pipeline?

An example sequence

- Processor core is right beside the L1 cache. Let, an instruction has to be processed:
  - Processor core fetches a long and complex x86 instruction from the L1 instruction cache
  - The instruction is sent into the pipeline where it is broken down into smaller units
  - If it is an integer operation, it is sent to an ALU, while floating point operations are sent to an FPU
  - After processing the data is sent back to the L1 cache
  - This description applies to the working cycle in, for example, the Pentium III and Athlon

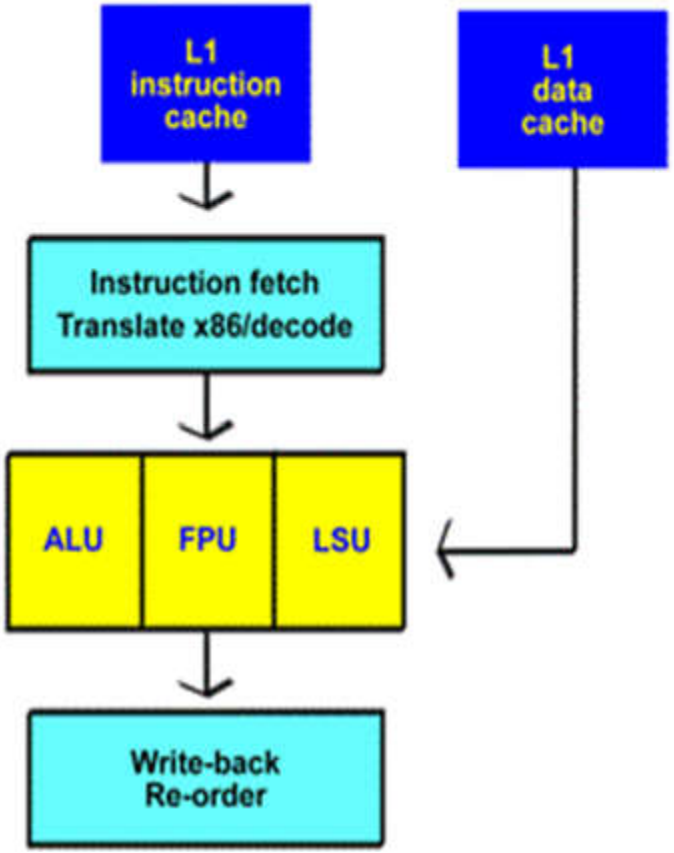


Figure 11 The passage of instructions through the pipeline

Pipelines (Cont')
What is actually happens in the pipeline?

- In Pentium 4, the instruction cache has been placed between the "Instruction fetch/Translate" unit and the ALU/FPU/LSU
- Here, the instruction cache doesn't store the actual instructions but rather the "half-digested" micro-ops

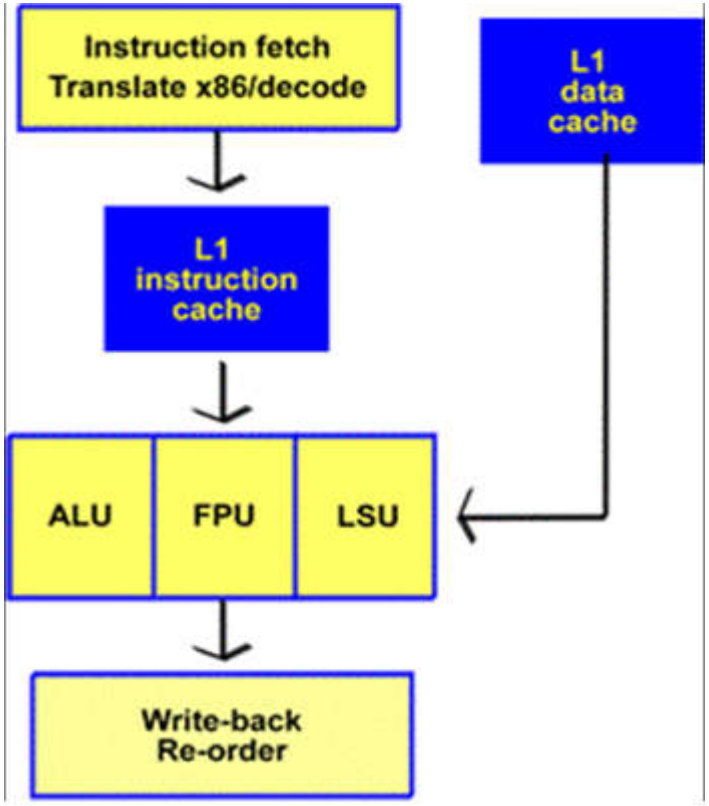


Figure 12 In the Pentium 4, the instruction cache stores decoded micro instructions

Pipelines (Cont')
Length of the pipe

- The longer the pipeline, the higher the processor's clock frequency can be
  - This is because in the longer pipelines, the instructions are cut into more (and hence smaller) sub-instructions which can be executed more quickly

CPU	Number of pipeline stages	Maximum clock frequency
Pentium	5	300 MHz
Motorola G4	4	500 MHz
Motorola G4e	7	1000 MHz
Pentium II and III	12	1400 MHz
Athlon XP	10/15	2500 MHz
Athlon 64	12/17	>3000 MHz
Pentium 4	20	>3000 MHz
Pentium 4 "Prescott"	31	>5000 MHz

Pipelines (Cont')

Problems of having more Pipelines

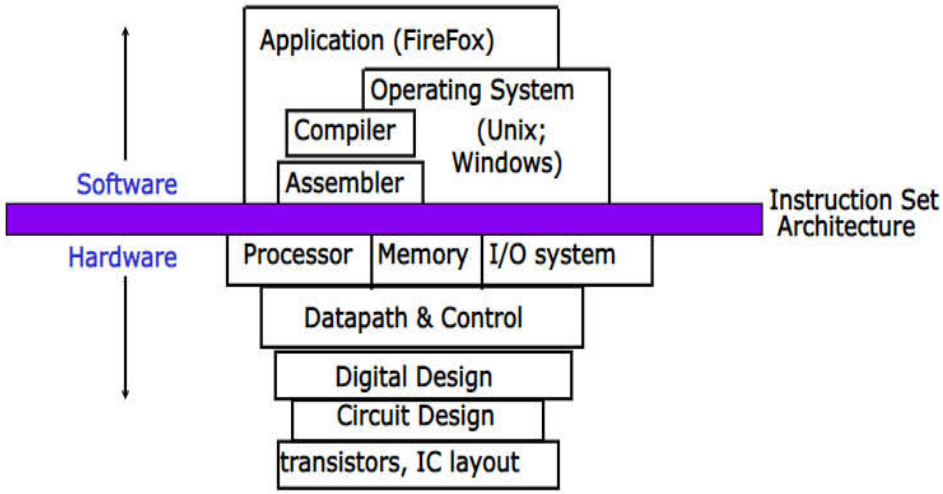
- More and more parallel pipelines in the one CPU is not the solution to improve the performance to highest level
- The processor core is simply not utilized efficiently enough, because data cannot be brought to it quickly enough
- Another problem of having several pipelines arises when the processor can decode several instructions in parallel – each in its own pipeline
  - *Mis-prediction* and results in a number of wasted clock ticks
    - ✓Solution by Intel: *Branch Prediction Unit*, which constantly attempts to guess the correct instruction sequence

Operation (cont')

- **Decode**

- In this step, CPU determines what the CPU will do with the fetching instruction
- Decoding performed by the circuitry known as the *instruction decoder*
  - Instruction is converted into signals that control other parts of the CPU

- The way in which the instruction is interpreted is defined by the CPU's instruction set architecture (ISA)
  - ISA, of a computer is an interface between the hardware and lowest level software



Instructions and data

- An instruction is an elementary operation that the processor can accomplish
- An instruction has two fields:
  - Operation code – represents the action that the processor must execute
  - Operand code – defines the parameters of the action
- The number of bits in an instruction varies according to the type of data
- Instructions can be grouped by category, of which the main ones are:
  - **Memory Access:** accessing the memory or transferring data between registers
  - **Arithmetic Operations:** operations such as addition, subtraction, division or multiplication
  - **Logic Operations:** operations such as AND, OR, NOT, EXCLUSIVE NOT, etc.
  - **Control**: sequence controls, conditional connections, etc.

Instructions and data (Cont')

- Data is typically user data (example: email writing)
  - The actual contents (i.e. text, letters) are user data
  - End user uses software to send program code (instructions) to the processor

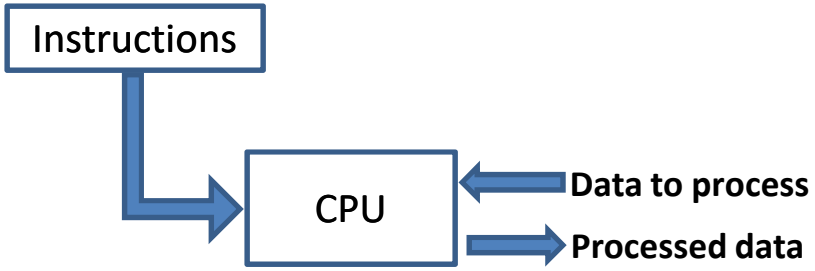
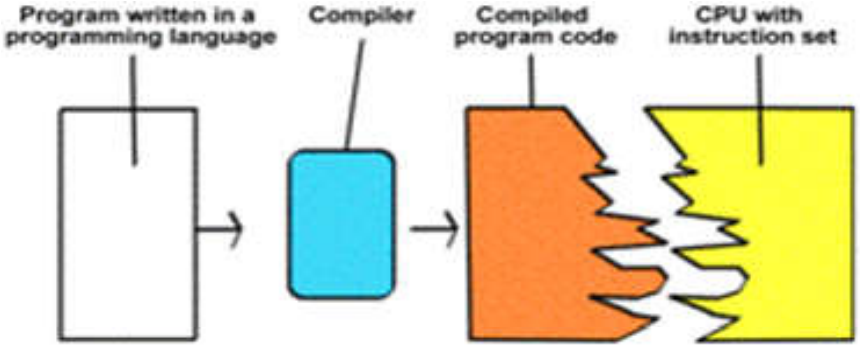


Figure 1 The instructions process the user data

Instructions and data (cont')

Instruction and Compatibility

- The program code produced has to match the CPU's instruction set otherwise it cannot be run



- The processors from AMD and Intel are compatible means they understand the same instructions
- Two different processors process the instructions internally can be different but externally all basically function same way

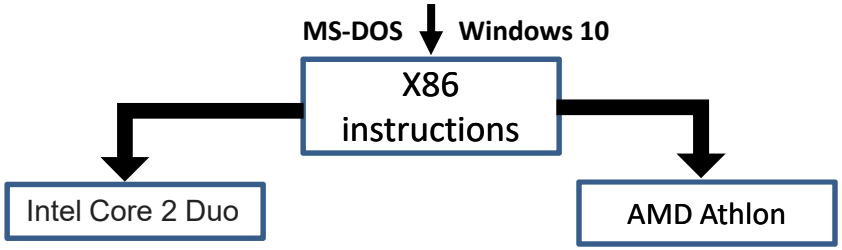


Figure 2 The x86 instruction set is common to all PC's

Instructions and data (cont')

X86 and CISC

- **X86 Instruction set designed for Intel 8086 processor (29000 transistor) is of the CISC (Complex Instruction Set Computer) type**
  - Instructions are diverse and complex
  - individual instruction vary in length
- **RISC (Reduced Instruction Set Computer)**
  - instructions all have the same length
  - Faster than CISC

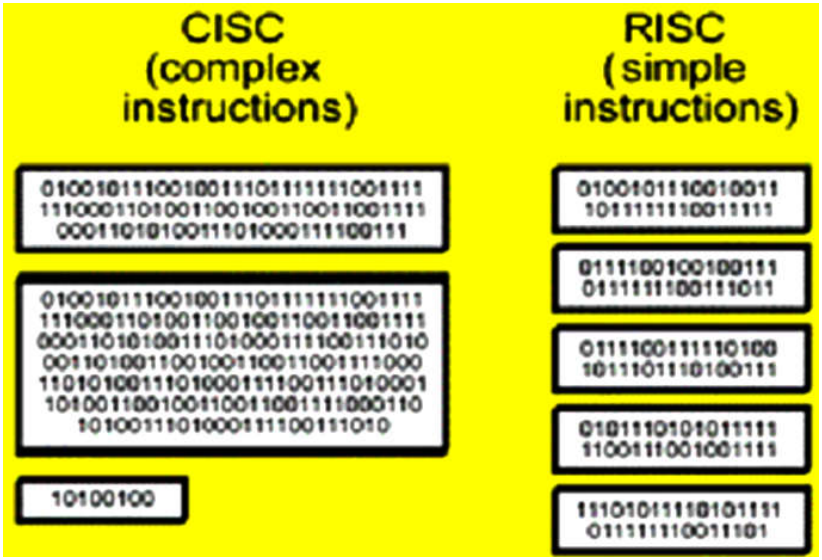


Figure 4 Mixture of CISC and RISC used by Pentium 4 and AthlonXP

Instructions and data (cont')

CISC vs. RISC

Keyword	CISC	RISC
Emphasis on	Hardware	Software
Instruction types	Complex	Simple
Number of Instructions	Extended (100-200)	Reduced (30-40)
Duration of an instruction	More cycles (4-120)	One cycle
Instruction format	Variable	Fixed
Instructions execution	Sequential	In parallel (Pipeline)
Addressing modes	complex	Simple
Register set	unique	Multiple

Instructions and data (cont')

X86 and CISC (cont')

- In order to maintain compatibility with the older DOS/Windows programs, the later CPU's still understand CISC instructions
- They are just converted to shorter, more RISC-like, sub-operations (called micro-ops), before being executed
- Most CISC instructions can be converted into 2-3 micro-ops

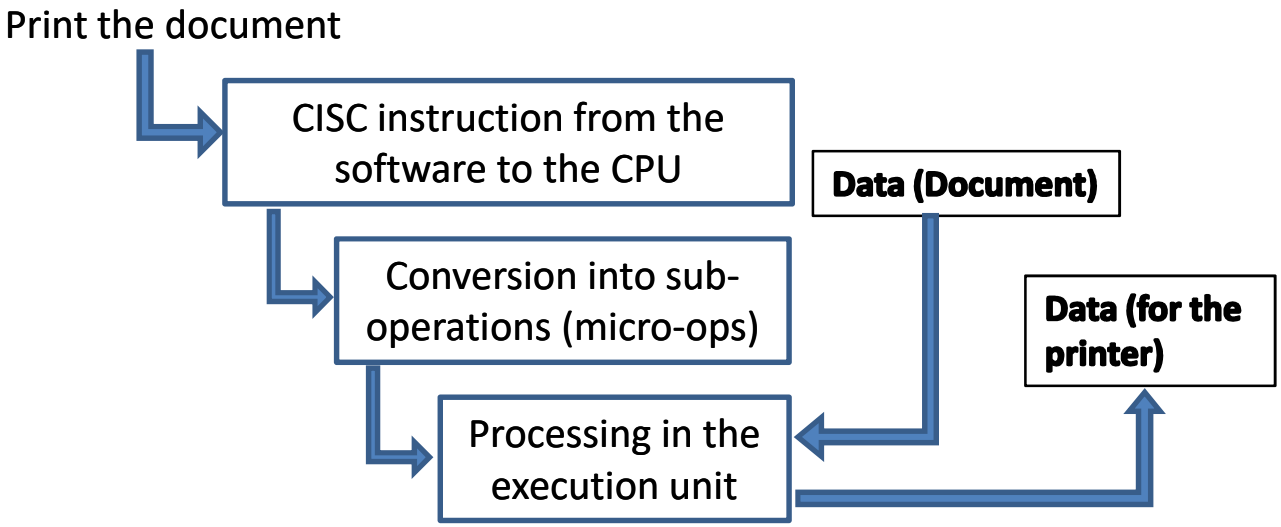


Figure 5 The CISC instructions are decoded before being executed in a modern processor. This preserves compatibility with older software

Instructions and data (cont')

Extensions to the instruction set

- For each new generation of CPU's, the original instruction set has been extended
  - 80386 processor added 26, 80486 added six, Pentium added eight new instructions
- At the same time, execution of the instructions was made more efficient
  - To add one number to a running summation, 80386 processor needs six clock ticks, where 80486 takes just two clock ticks
- Since then, MMX (*MultiMedia eXtension*, *Multiple Math eXtension*, or *Matrix Math eXtension)* and SSE (*S*treaming *S*IMD *E*xtensions) extensions have followed

Instructions and data (cont')

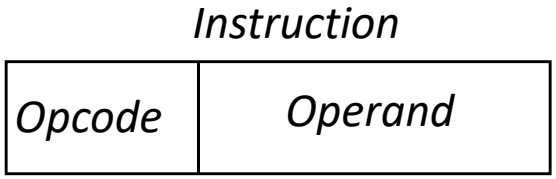
Extensions to the instruction set (cont')

- MMX is a single instruction, multiple data (SIMD) instruction set designed by Intel, introduced in 1997 with its P5-based Pentium line of microprocessors, designated as "Pentium with MMX Technology"
  - It developed out of a similar unit introduced on the Intel i860, and earlier the Intel i750 video pixel processor
  - MMX is a processor supplementary capability that is supported on recent IA-32 processors by Intel and other vendors
- Streaming SIMD Extensions (SSE) is an SIMD instruction set extension to the x86 architecture, designed by Intel and introduced in 1999 in their Pentium III series processors
  - SSE contains 70 new instructions, most of which work on single precision floating point data
  - SIMD instructions can greatly increase performance when exactly the same operations are to be performed on multiple data objects
  - Typical applications are digital signal processing and graphics processing

Operation (cont')

• Decode (Cont')

- Machine language consists of strings of binary numbers



- Opcode is the first part of an instruction, which tells the computer what function to perform

- Operand is the second part of the instruction, which tells the computer where to find or store the data or instructions:

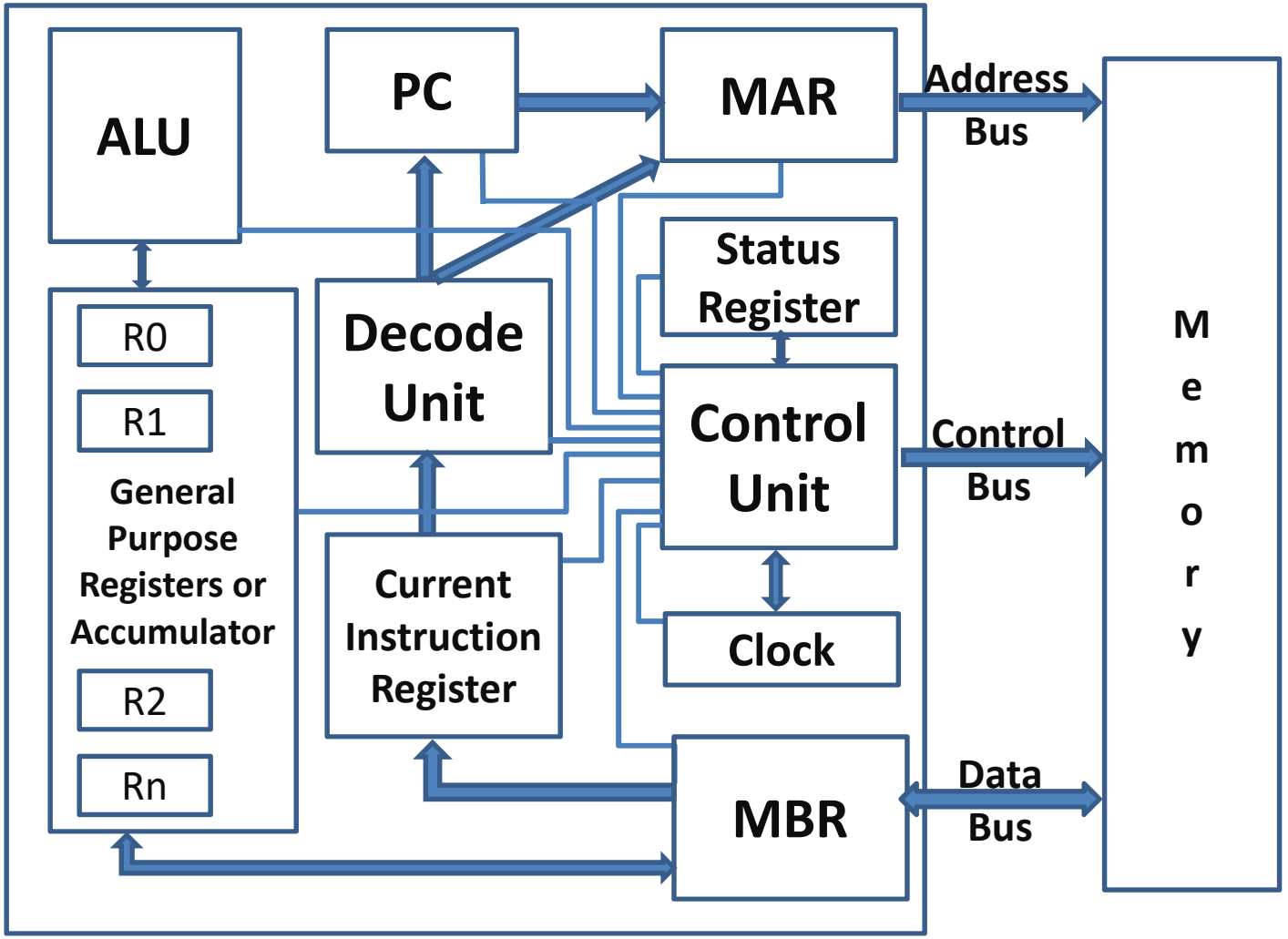
  - Operand could in the actual data or an address where the data is found

  - The number of operands (value, memory address or register) varies among computers. Each instruction tells the Control Unit of the CPU what to do and how to do.

  - Operations such as Arithmetic, Logical, etc.

Operation (cont')

Decode (Cont')

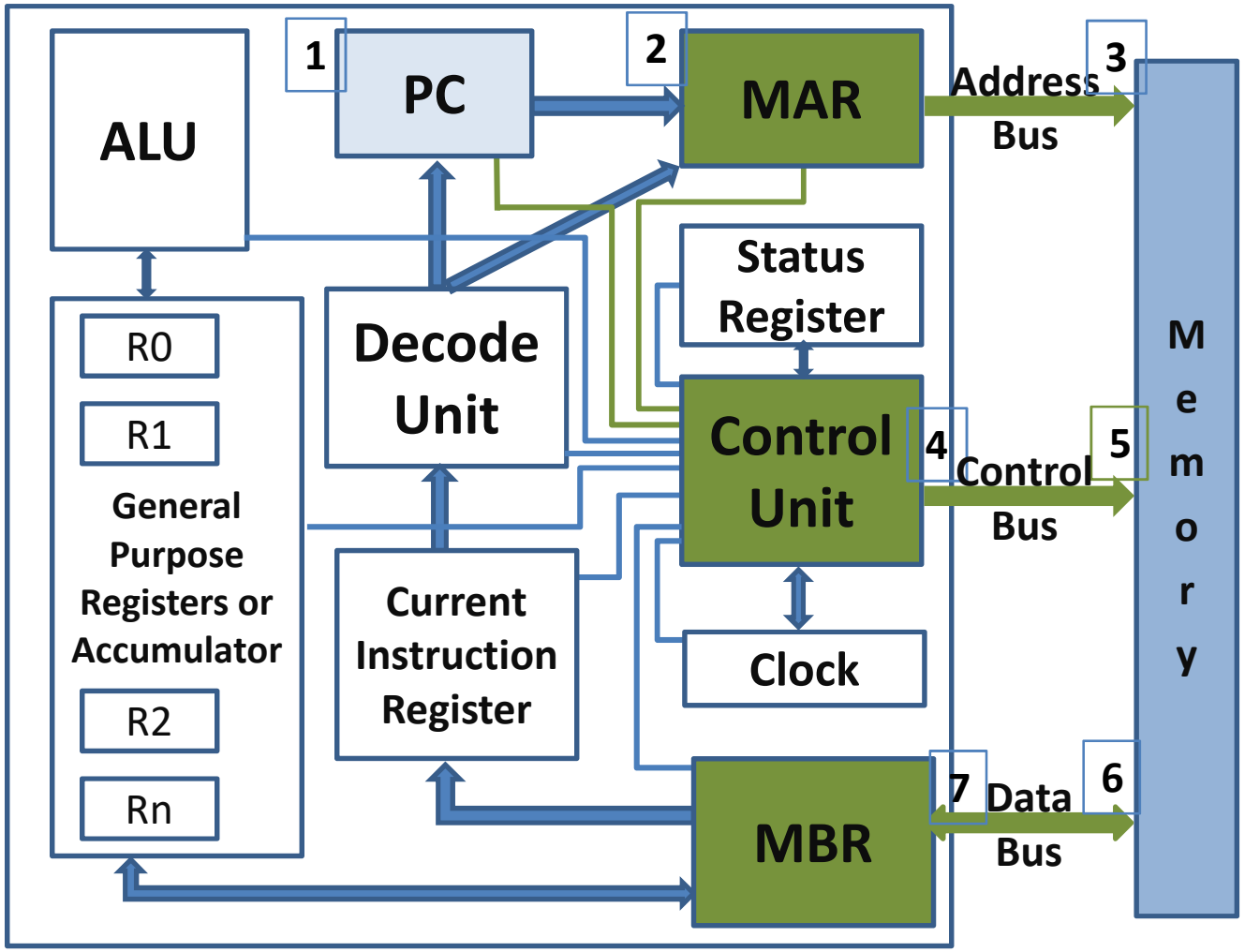


Operation (cont')

Decode (Cont')

Instructions: Fetch (Step by Step)

- **MAR (Memory Address Register) holds the address in memory which data or an instruction needs to be read from or written to.**

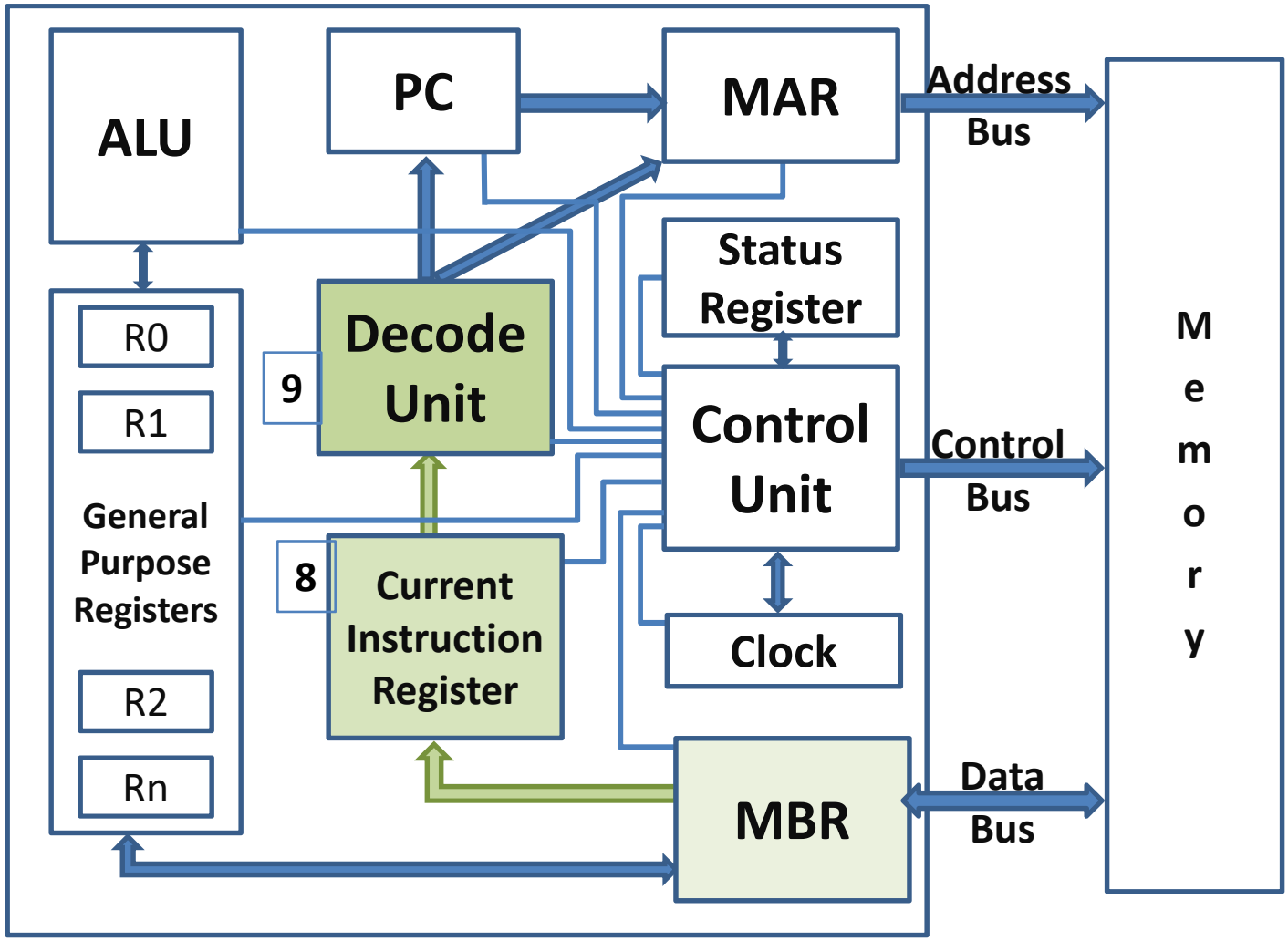


- **MDR (Memory Data Register) holds either data or an instruction which has been fetched from memory or is about to be written back to memory**

Operation (cont')

Decode (Cont')

Instructions: Decode (Step by Step)



- Instruction is copied into the CIR and then decode phase gets under way
- Break the instruction apart into its separate parts
- Instruction = Opcode + operand

Operation (cont')

Decode (Cont')

Instructions: Decode (Step by Step) [Cont']

Opcode	Instruction
0001	ADD
0010	SUB
0011	LDR
0101	CMP
0110	STR

Decode Unit

Control Unit

8bit instruction = Opcode + Operand

0	0	1	1	0	0	1	1

Basic machine operation

Addressing mode

Current Instruction Register

- Opcode means what to do (Operation) and Operand means what to do it to (Operand could contain actual data or an address where the data is found)
- Addressing mode incorporated into the bits allocated to the opcode
- Example:
  - 4-bits for opcode
    - 3-bits for basic machine operation
    - 1-bit for addressing mode
  - 4 bits for operand
- Total of 16 different opcodes available in this representation

Operation (cont')

Execute

- Execute step is performed next to the fetch and decode steps
    - This step consists of a single action or a sequence of actions
- Various parts of CPU are electrically connected to make each action operation successful
- Very often results are written to an internal register for quick access by subsequent instruction and other cases results may be written to slower main memory
- The entire process repeats with the next instruction cycle after the execution of an instruction

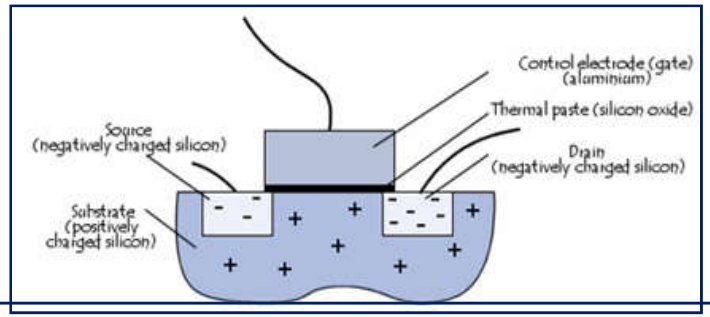
Transistor

• A transistor is **a binary switch and the fundamental building block of computer circuitry.** Like a light switch on the wall, the transistor either prevents or allows current to flow through. A single modern CPU can have hundreds of millions or even billions of transistors.

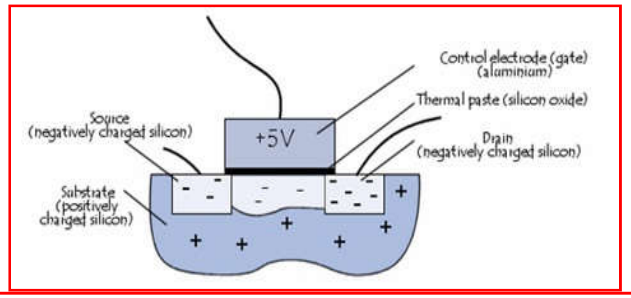
•A MOS (*metal, oxide, silicone*) transistor is the most common type of transistor used to design integrated circuits

• MOS transistors have two negatively charged areas: (i) Source – which has an almost zero charge), and (ii) Drain (which has a 5V charge), separated by a positively charged region, called a substrate

• The substrate has a control electrode overlaid, called a gate, that allows a charge to be applied to the substrate



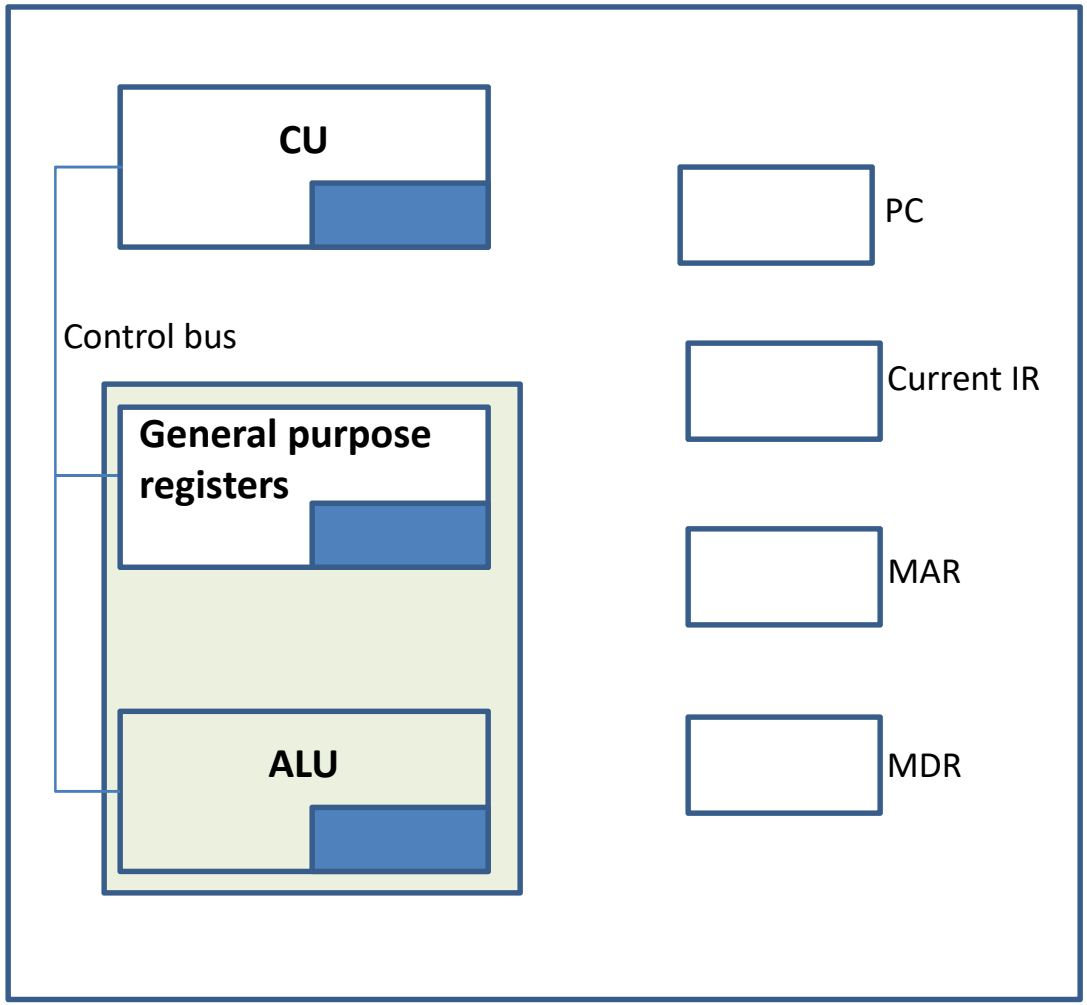
When there is no charge on the control electrode, the positively charged substrate acts as a barrier and prevents electron movement from the source to the drain.



When a charge is applied to the gate, positive charges of the substrate are repelled and a negatively charged communication channel is opened between the source and the drain

Operation (cont')

A detail example of Fetch-Decode-Execute

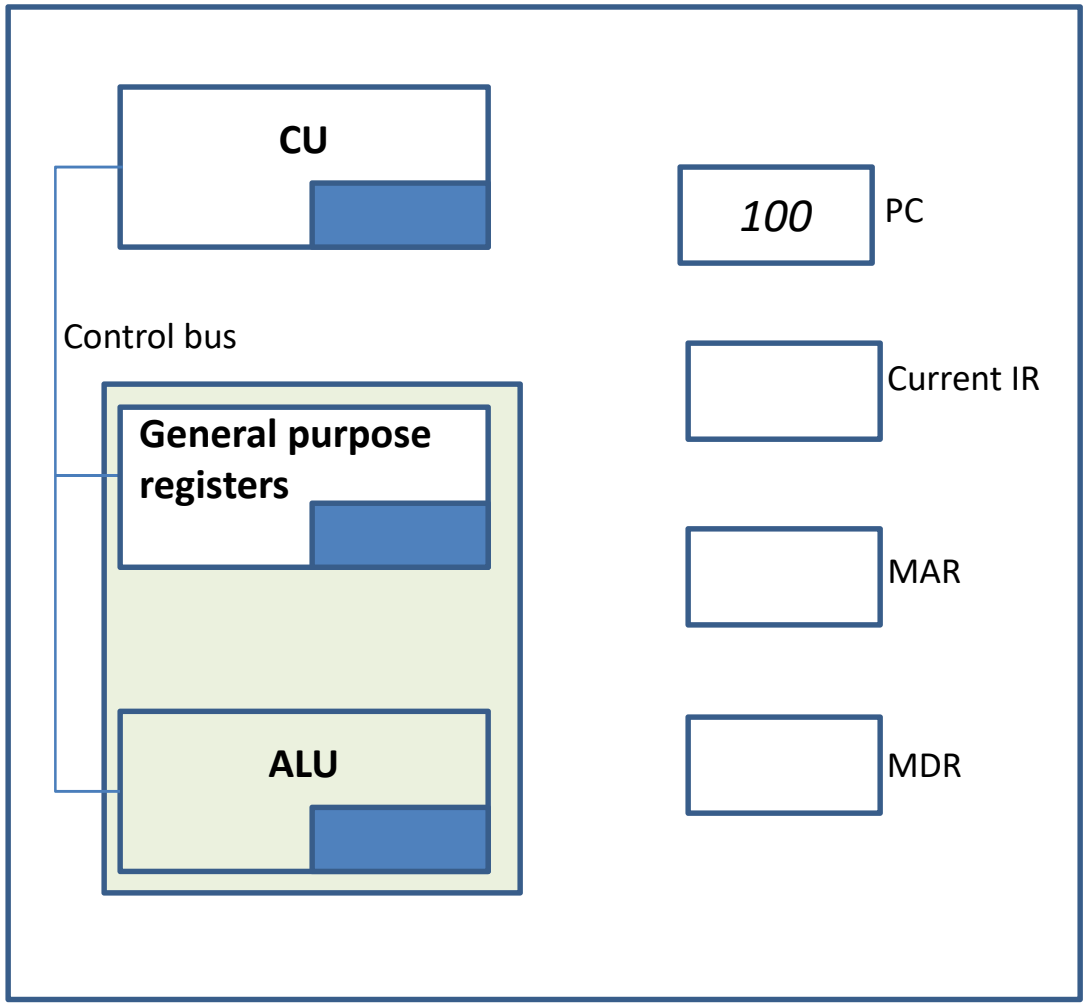


Memory

0	*9*
2	*10*
3	*11*
0	*12*
0	*13*
0	*14*
. . .	. . .
0	*99*
LOAD 10	*100*
ADD 11	*101*
STORE 12	*102*
0	*103*

Operation (cont')

A detail example of Fetch-Decode-Execute

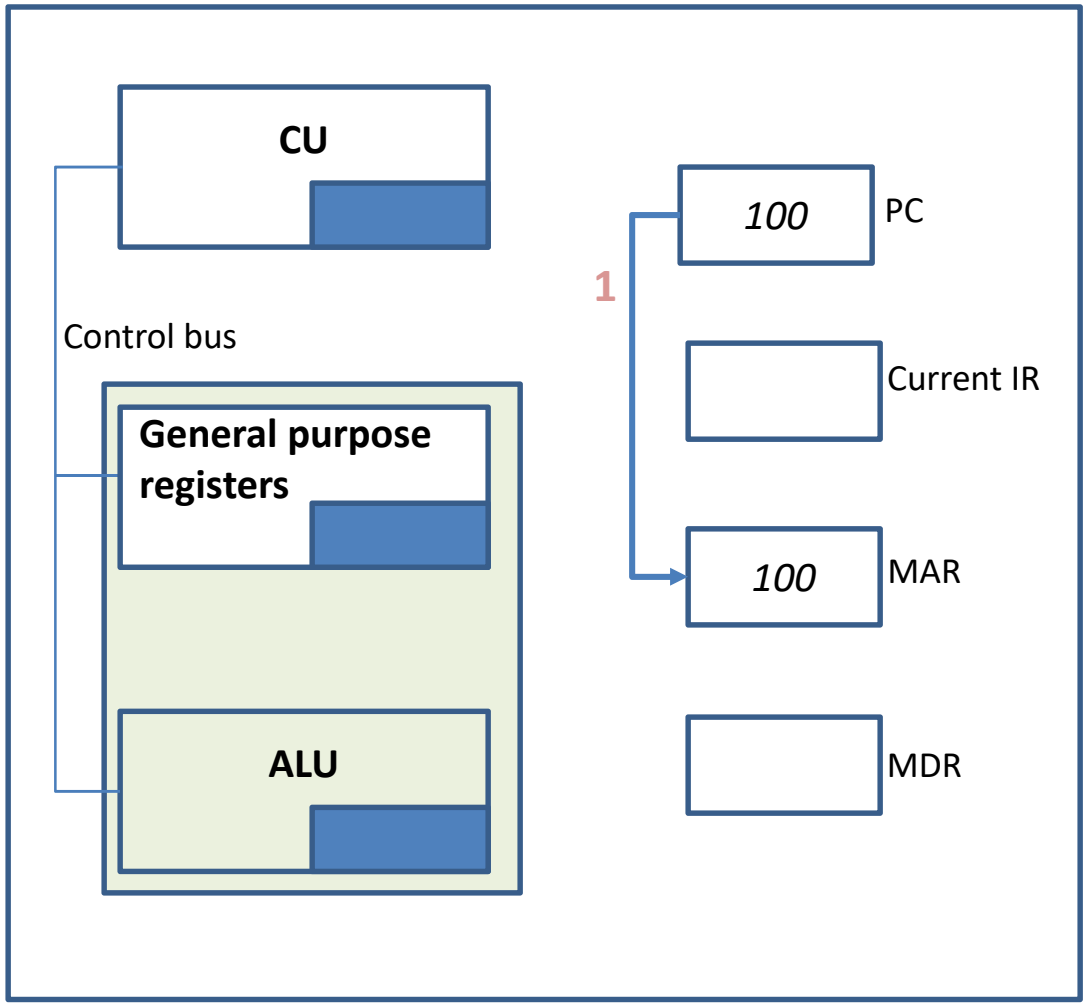


Memory

0	*9*
2	*10*
3	*11*
0	*12*
0	*13*
0	*14*
. . .	. . .
0	*99*
LOAD 10	*100*
ADD 11	*101*
STORE 12	*102*
0	*103*

Operation (cont')

A detail example of Fetch-Decode-Execute

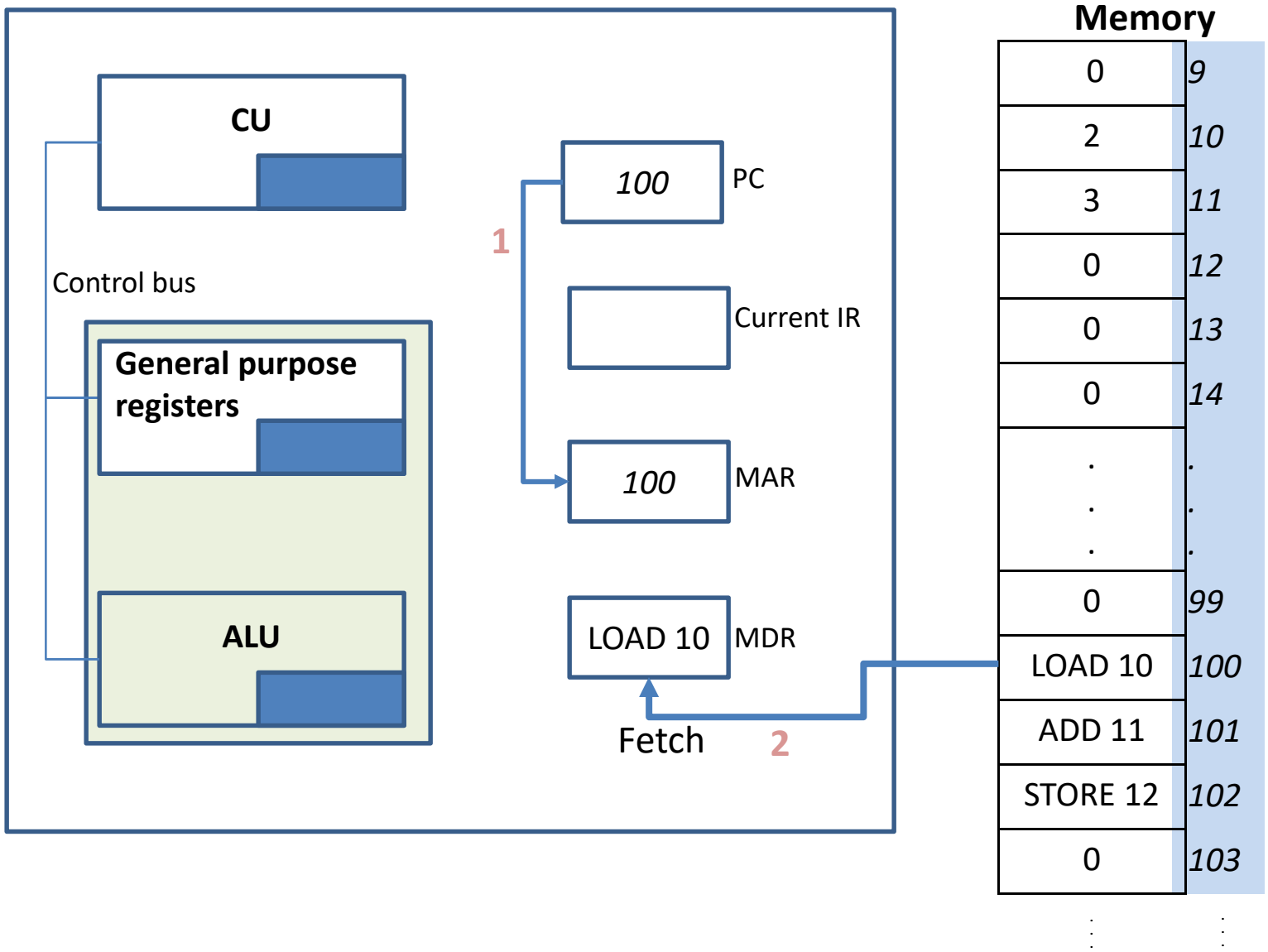


Memory

0	*9*
2	*10*
3	*11*
0	*12*
0	*13*
0	*14*
. . .	. . .
0	*99*
LOAD 10	*100*
ADD 11	*101*
STORE 12	*102*
0	*103*

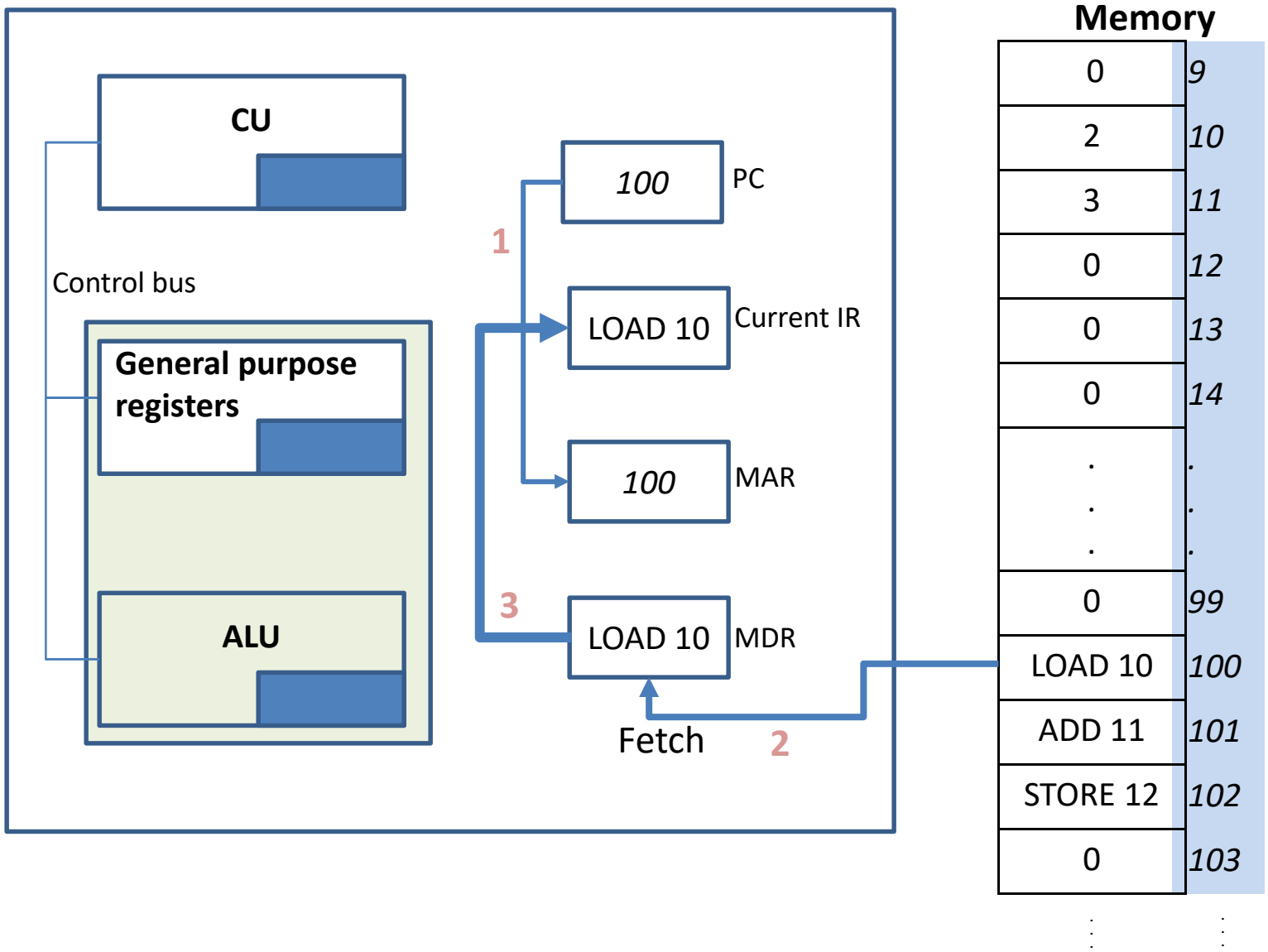
Operation (cont')

A detail example of Fetch-Decode-Execute



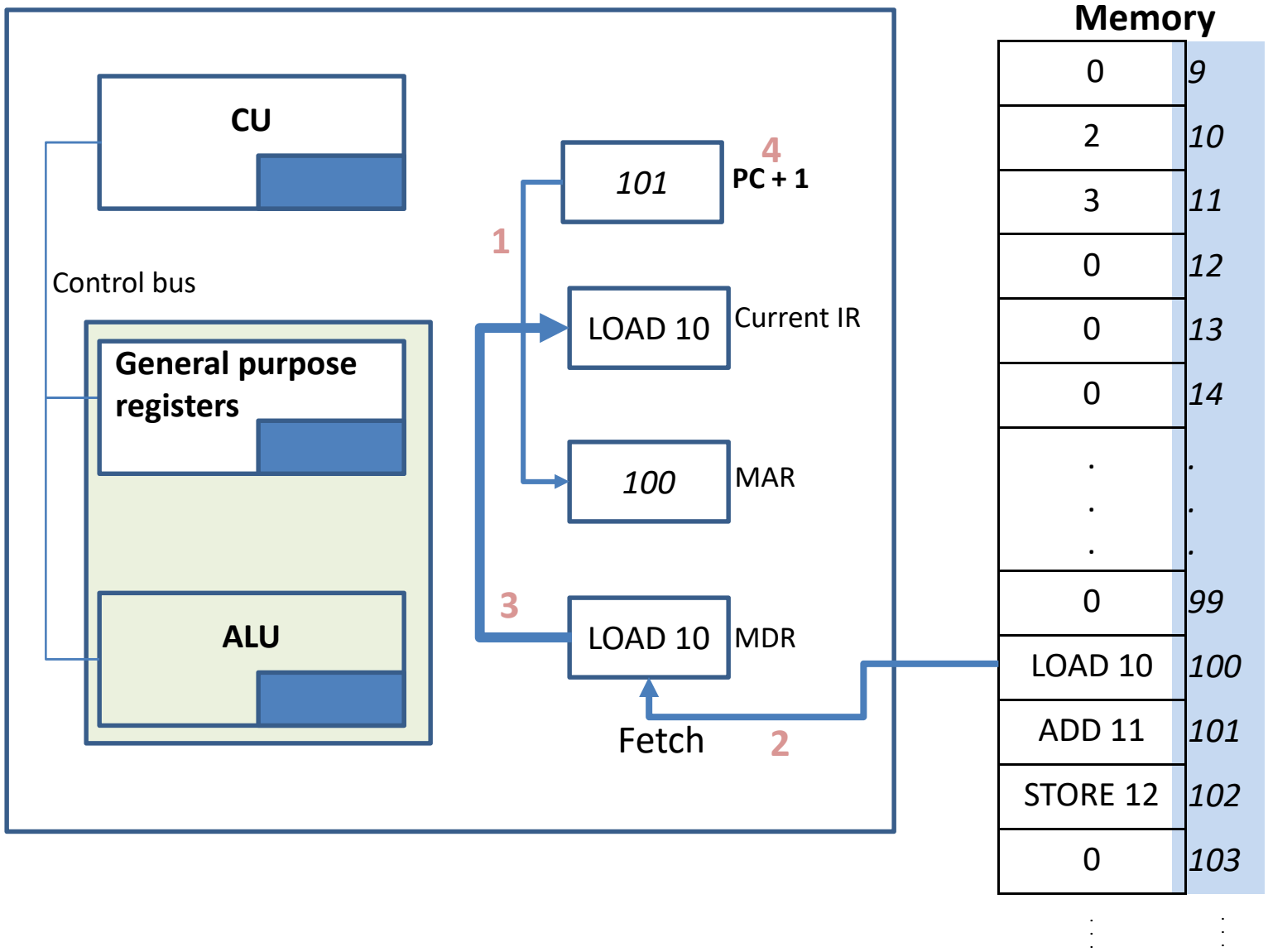
Operation (cont')

A detail example of Fetch-Decode-Execute



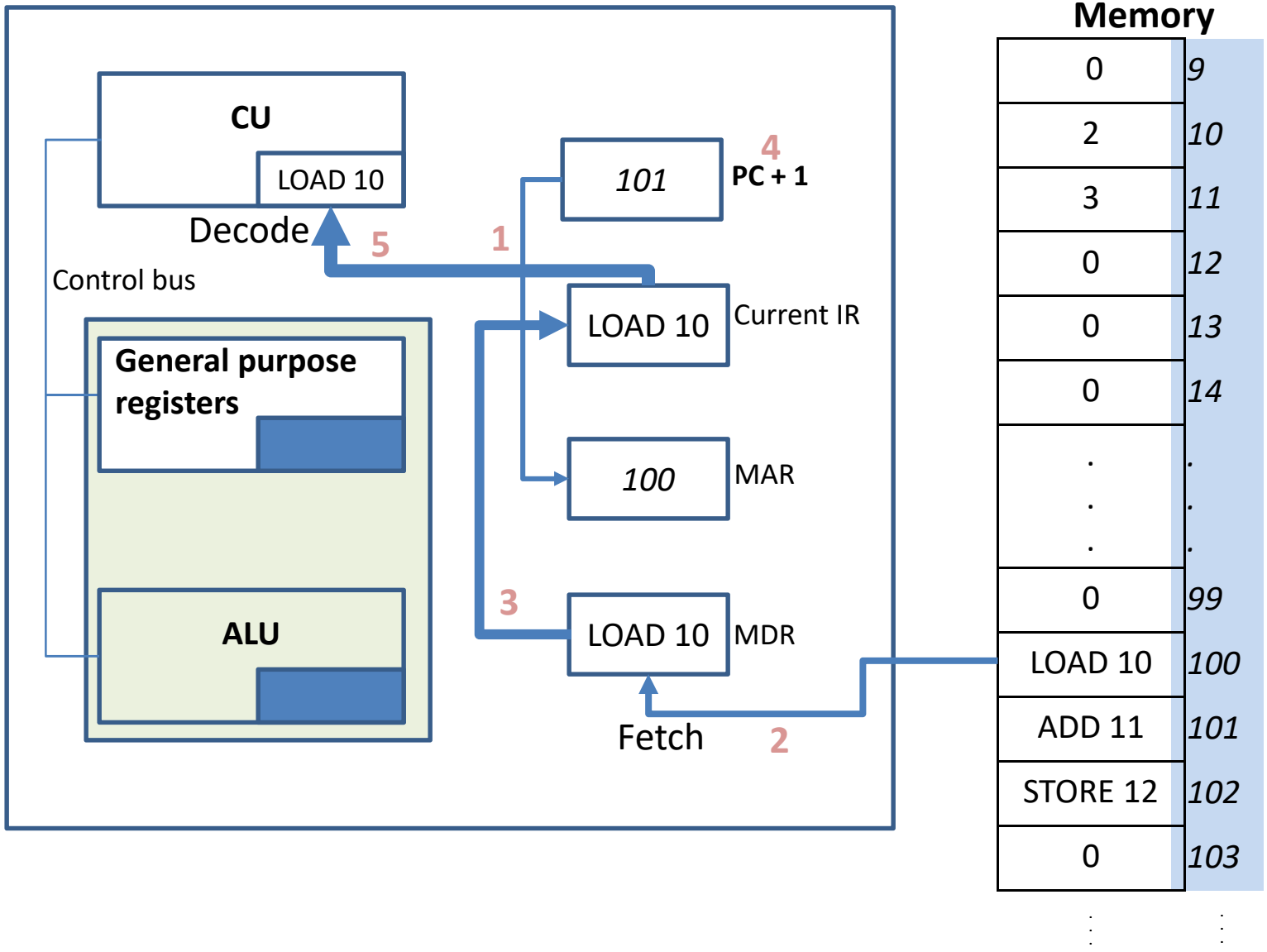
Operation (cont')

A detail example of Fetch-Decode-Execute



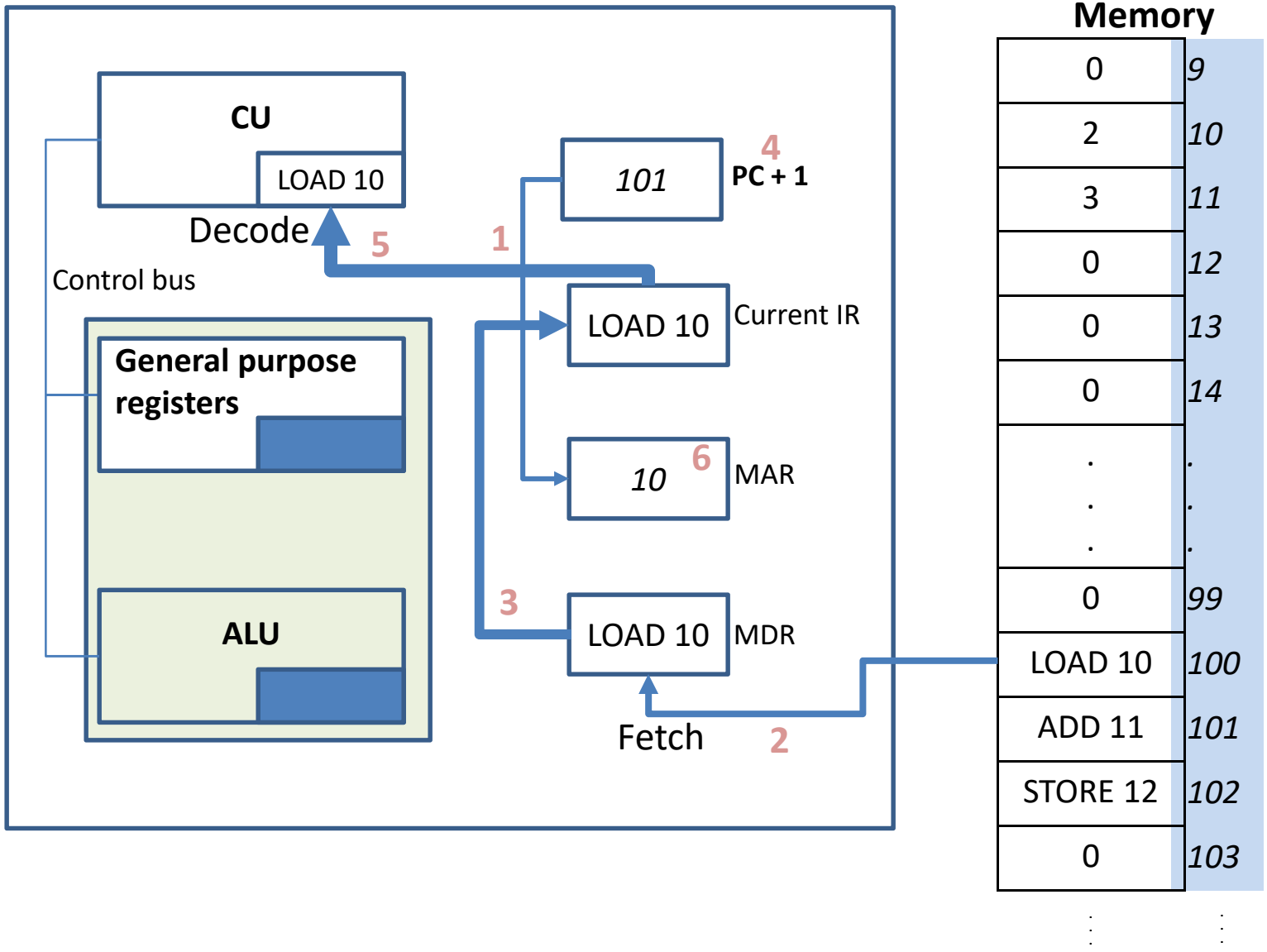
Operation (cont')

A detail example of Fetch-Decode-Execute



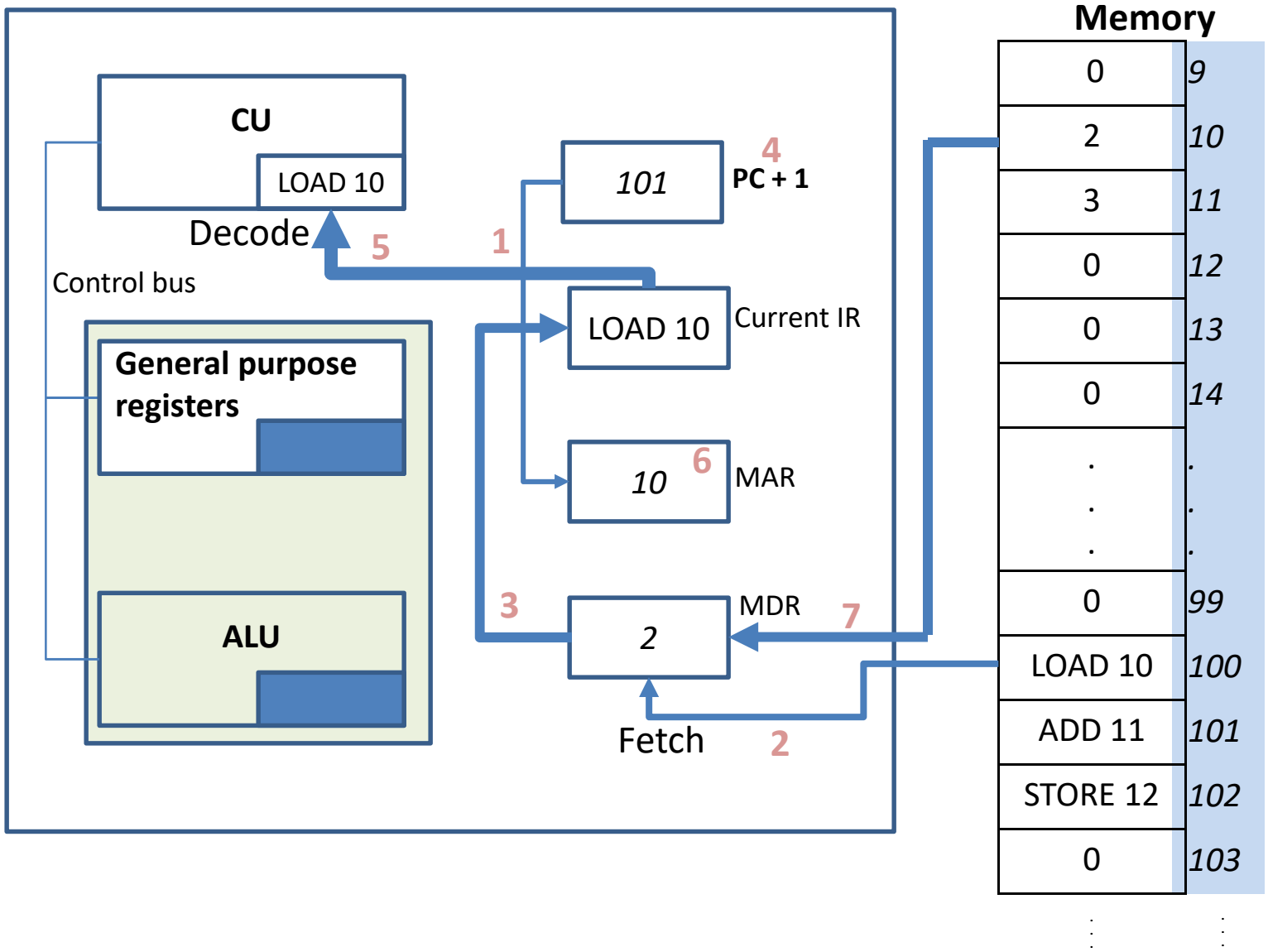
Operation (cont')

A detail example of Fetch-Decode-Execute



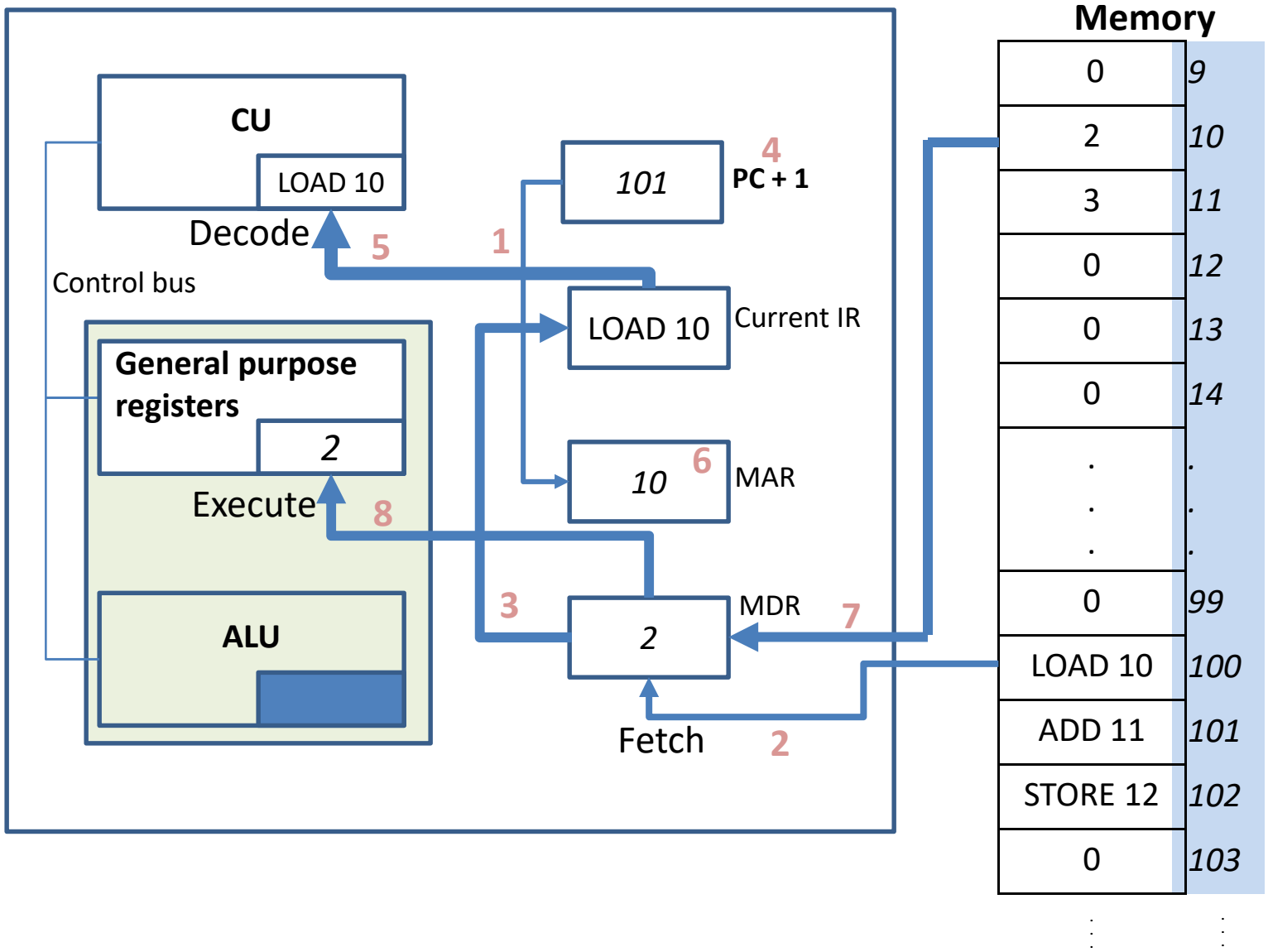
Operation (cont')

A detail example of Fetch-Decode-Execute



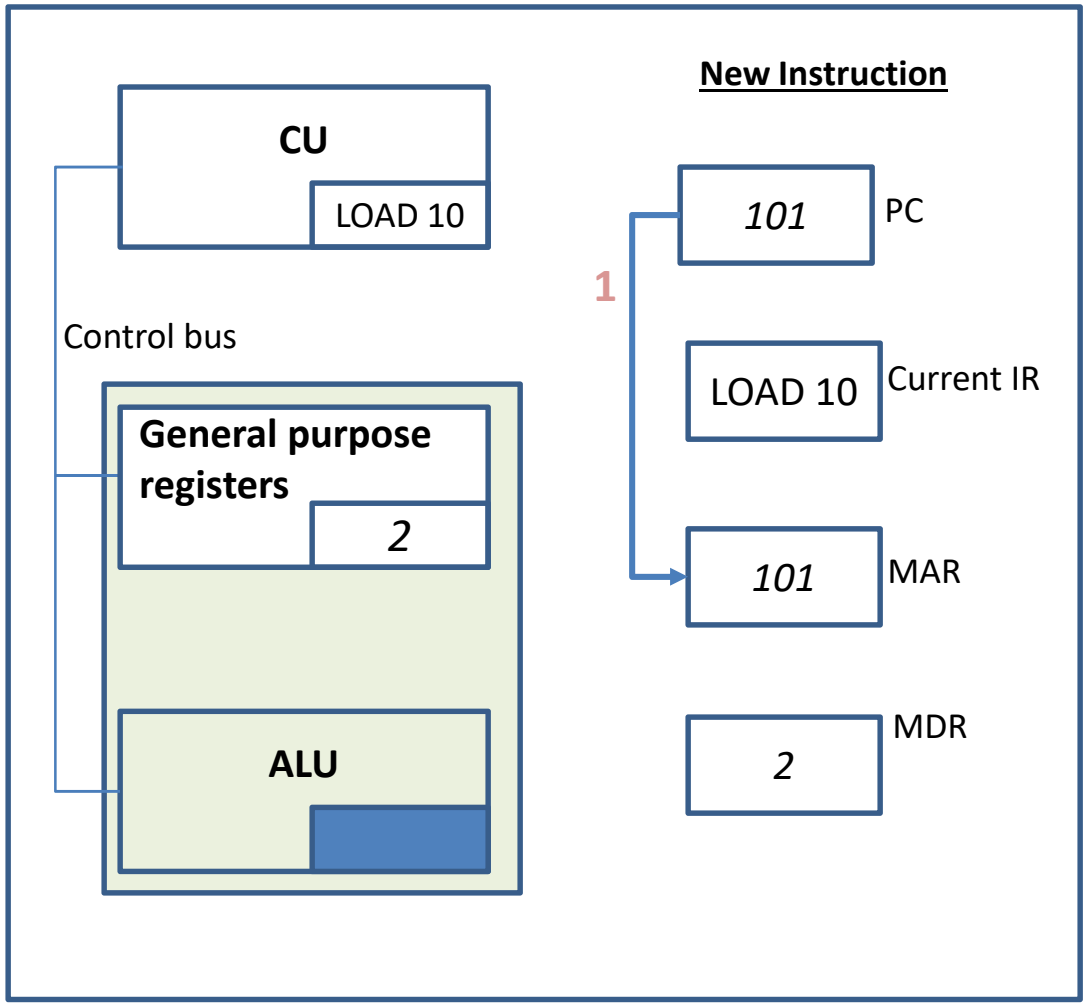
Operation (cont')

A detail example of Fetch-Decode-Execute



Operation (cont')

A detail example of Fetch-Decode-Execute

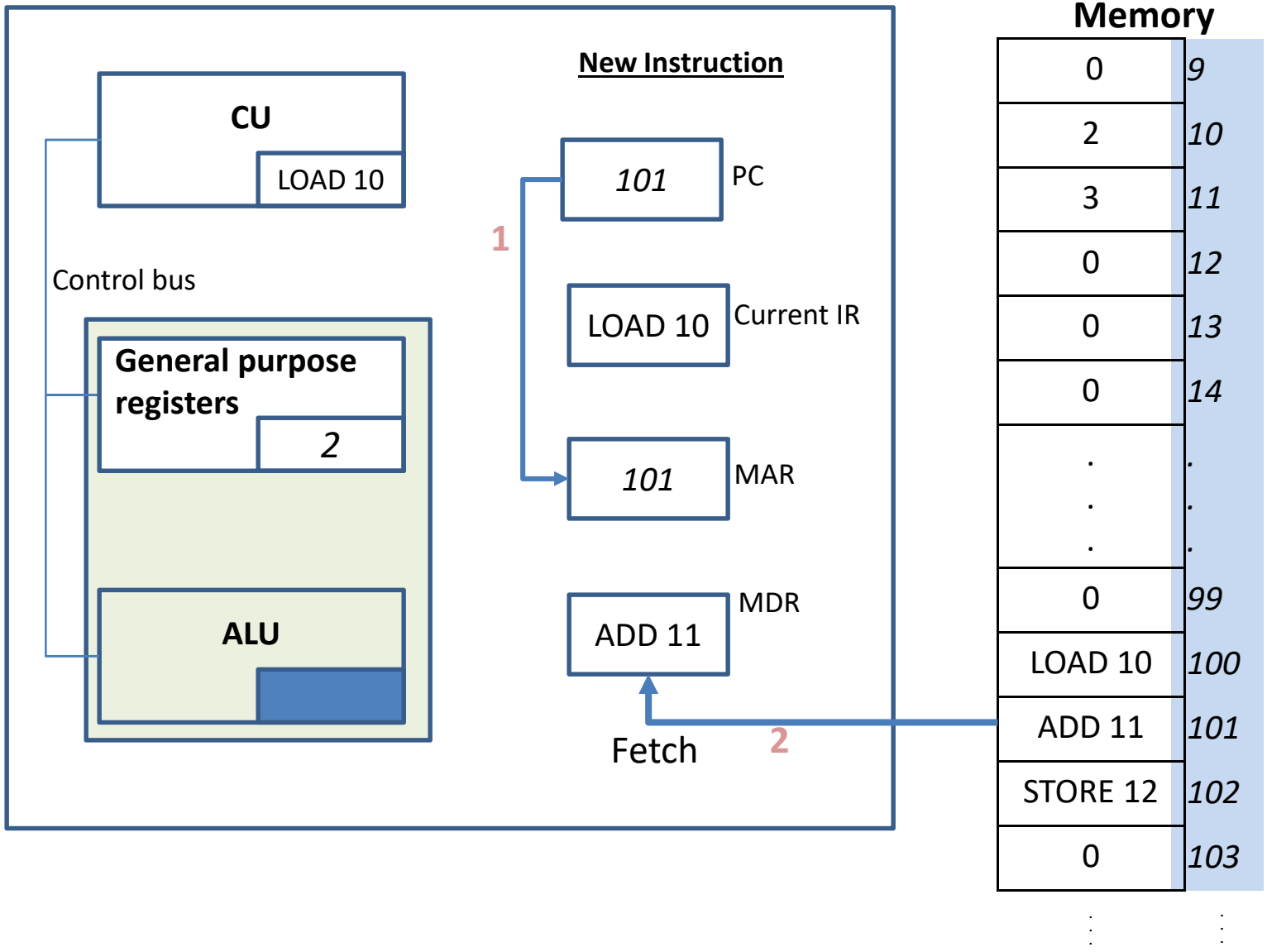


Memory

0	*9*
2	*10*
3	*11*
0	*12*
0	*13*
0	*14*
. . .	. . .
0	*99*
LOAD 10	*100*
ADD 11	*101*
STORE 12	*102*
0	*103*

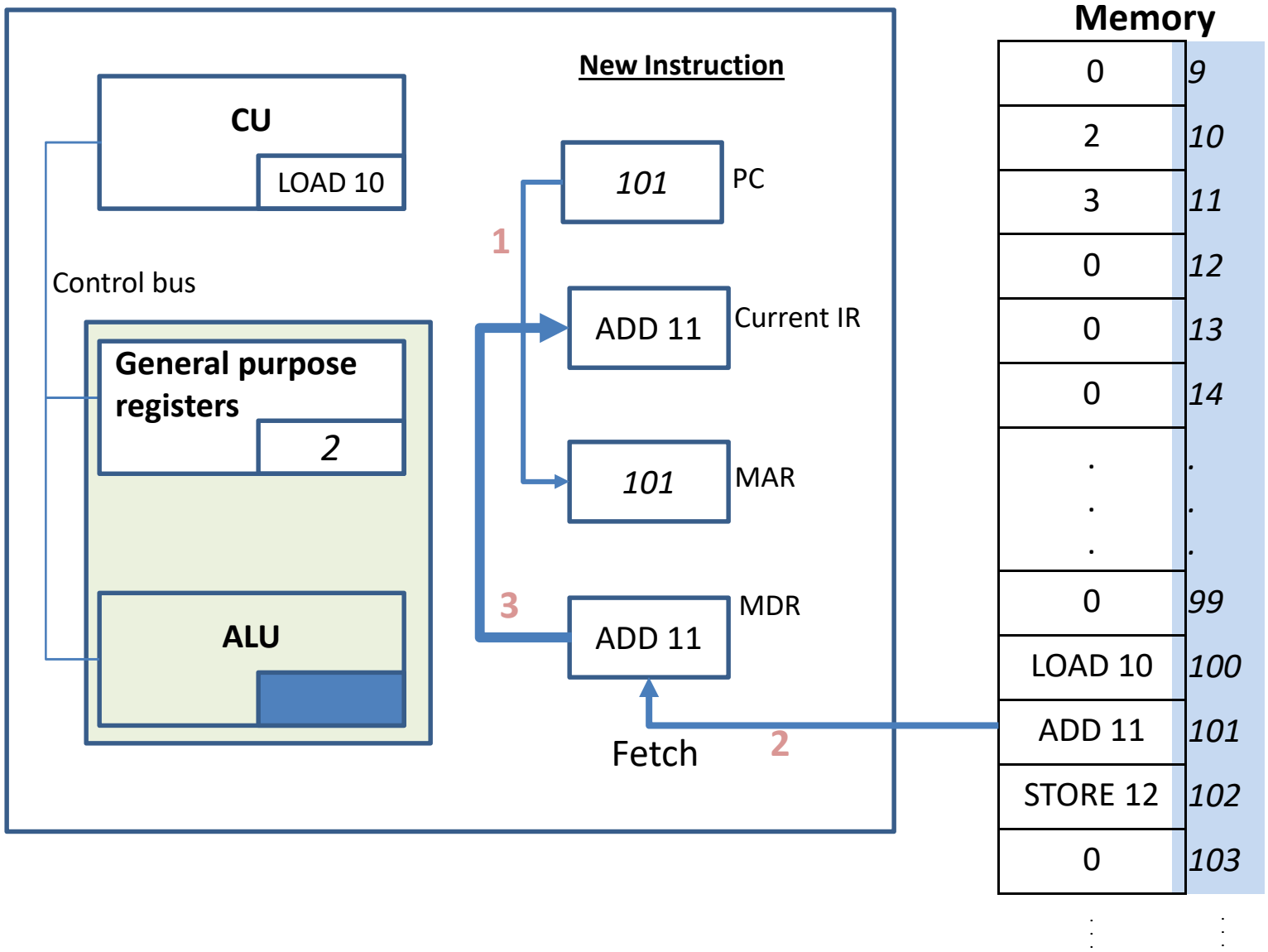
Operation (cont')

A detail example of Fetch-Decode-Execute



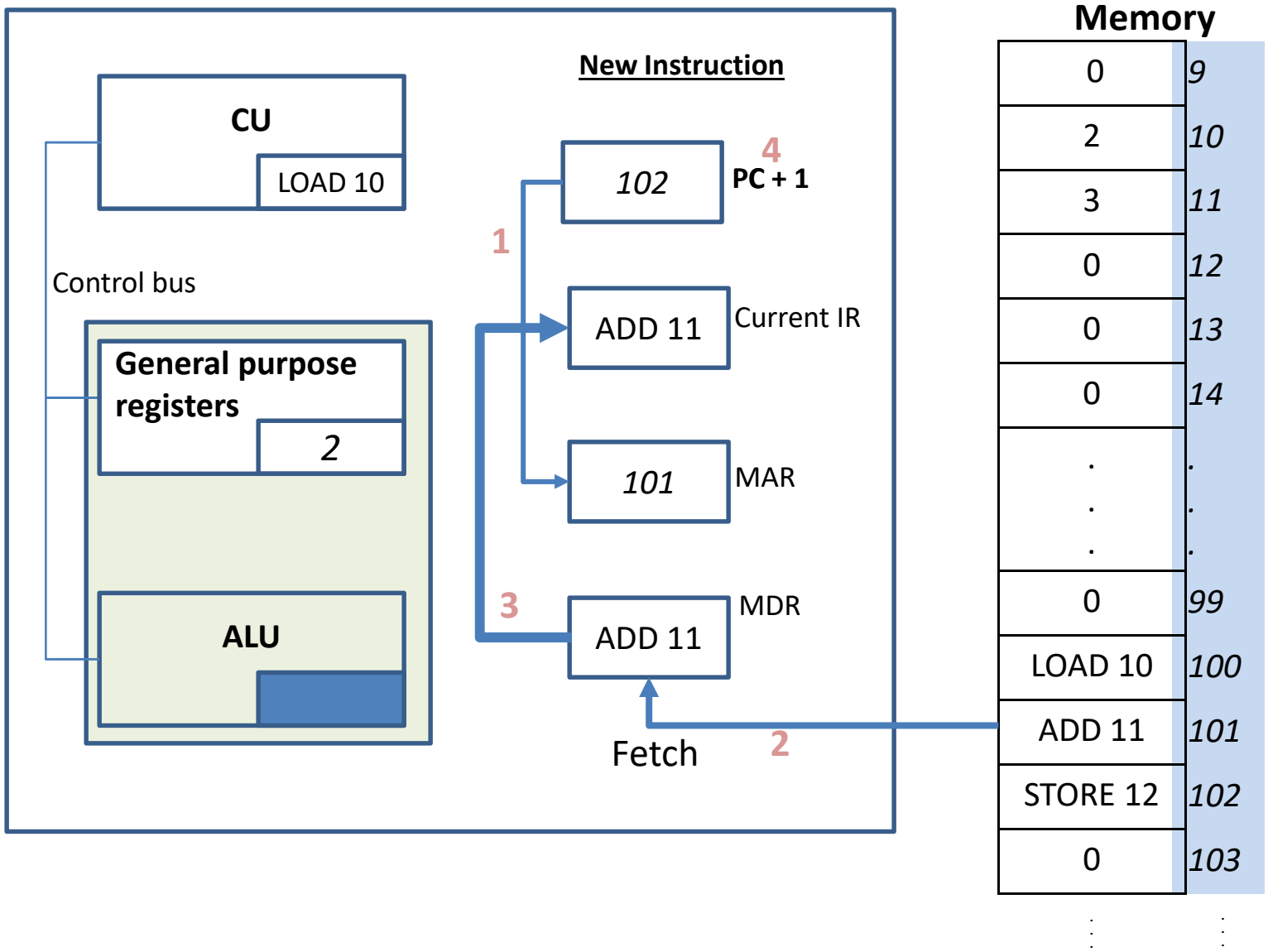
Operation (cont')

A detail example of Fetch-Decode-Execute



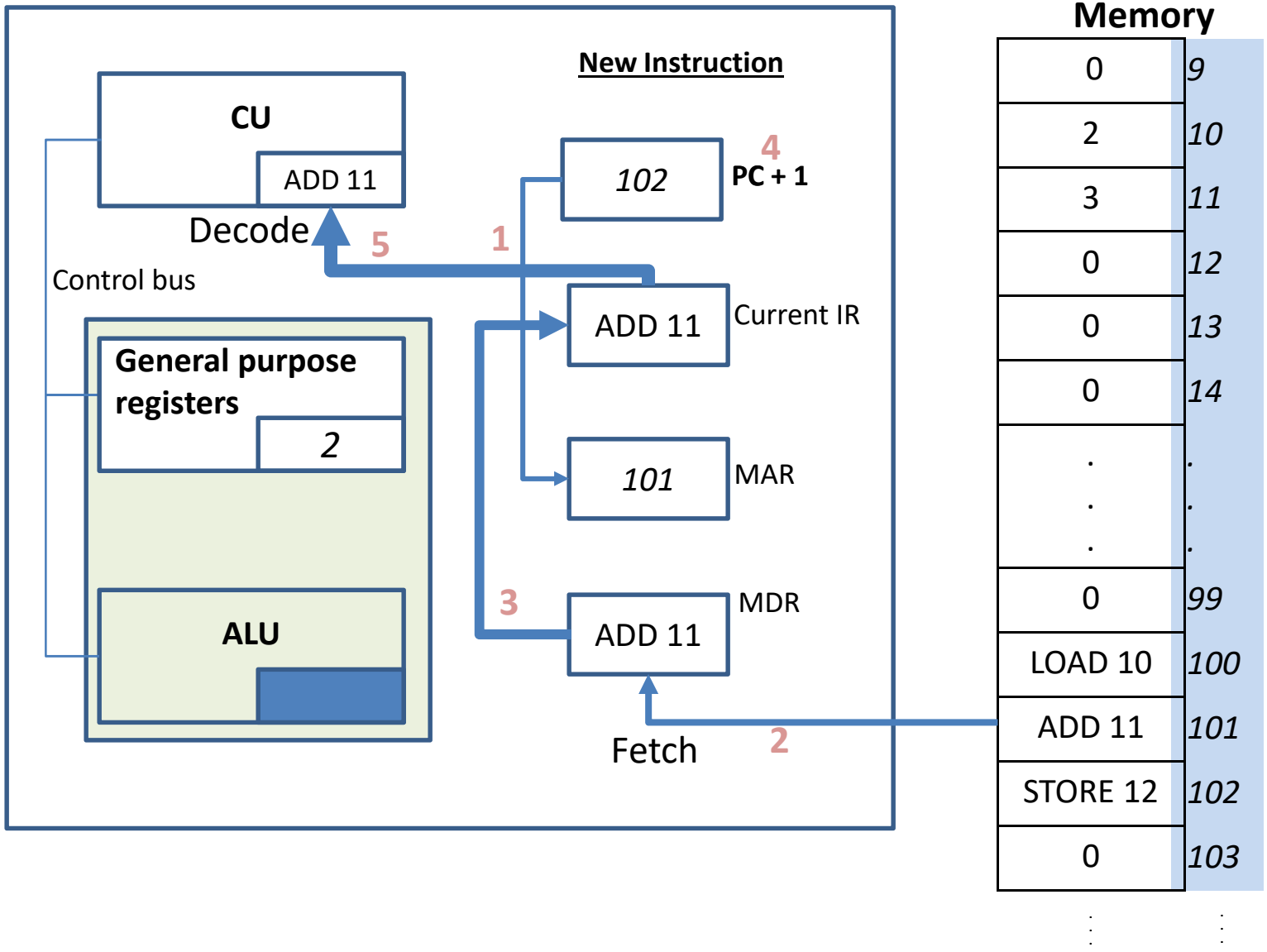
Operation (cont')

A detail example of Fetch-Decode-Execute



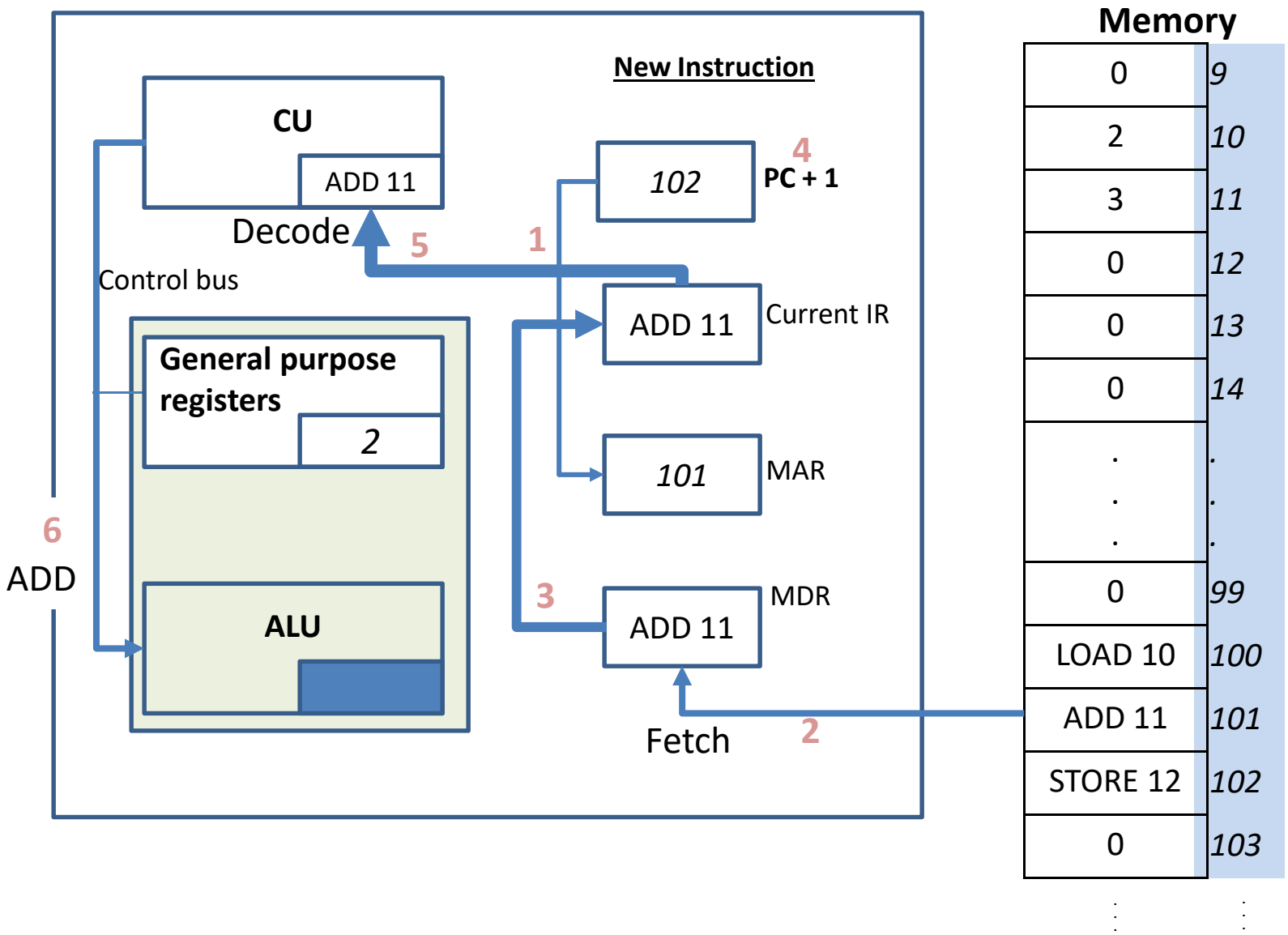
Operation (cont')

A detail example of Fetch-Decode-Execute



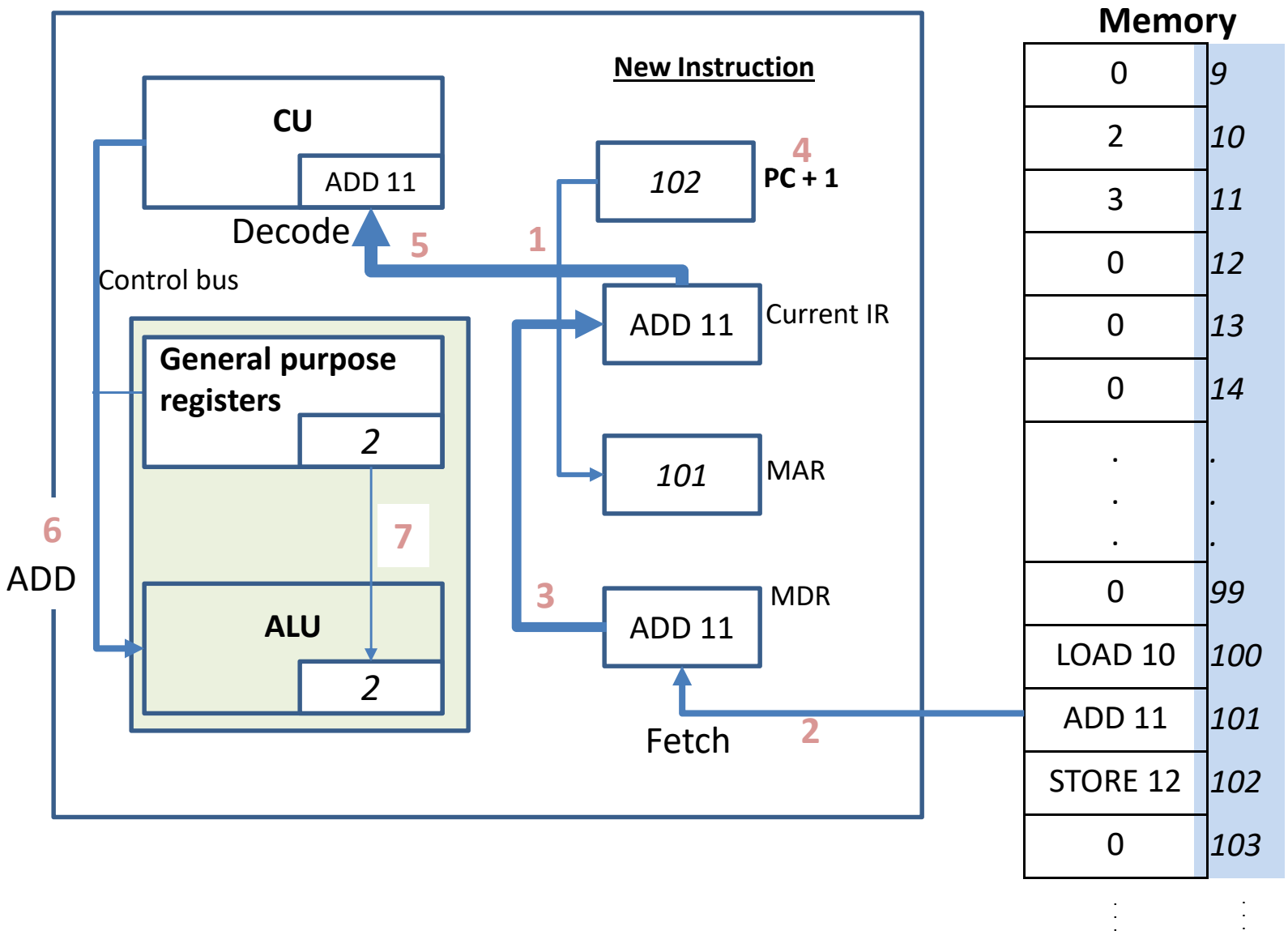
Operation (cont')

A detail example of Fetch-Decode-Execute



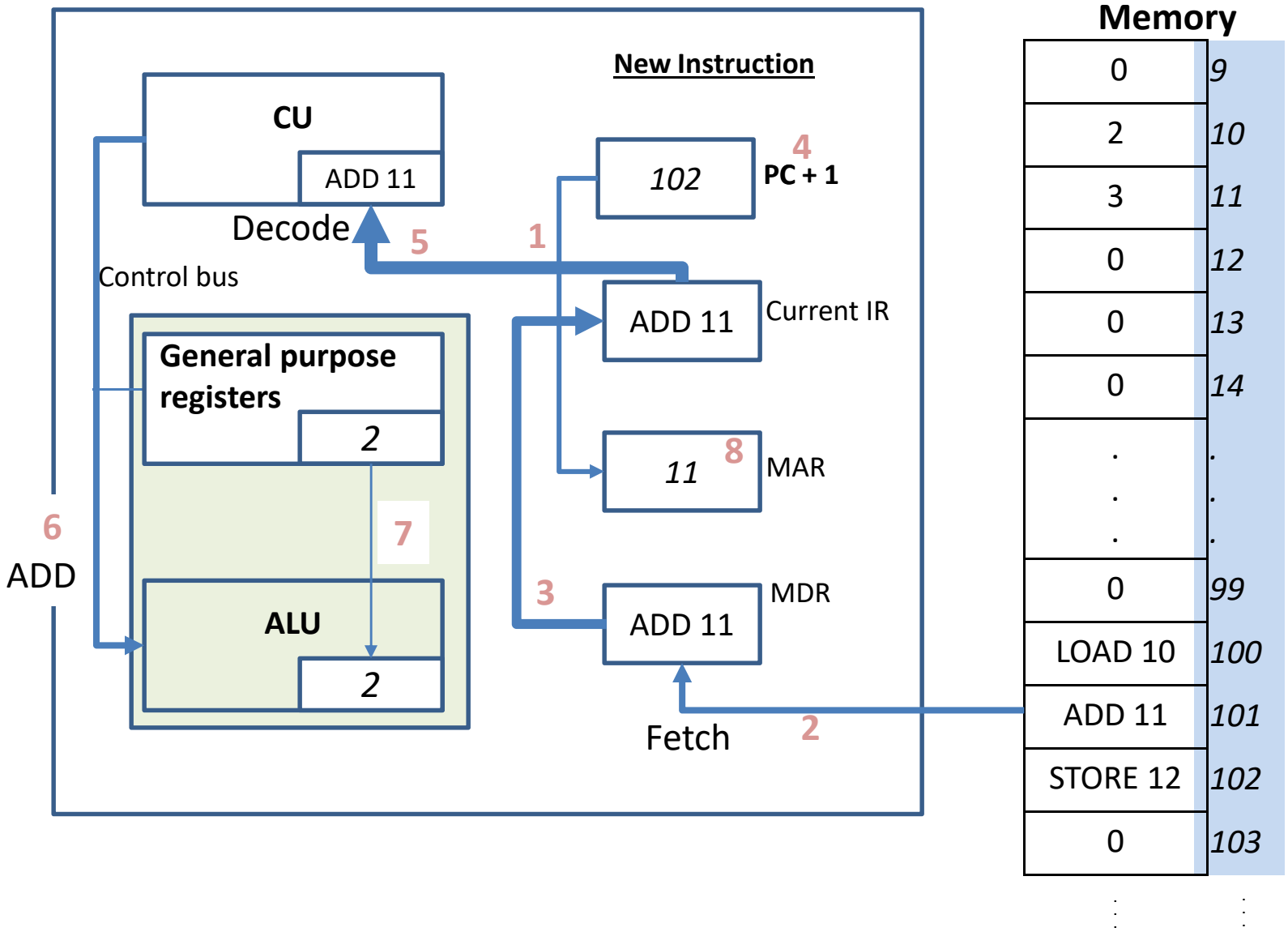
Operation (cont')

A detail example of Fetch-Decode-Execute



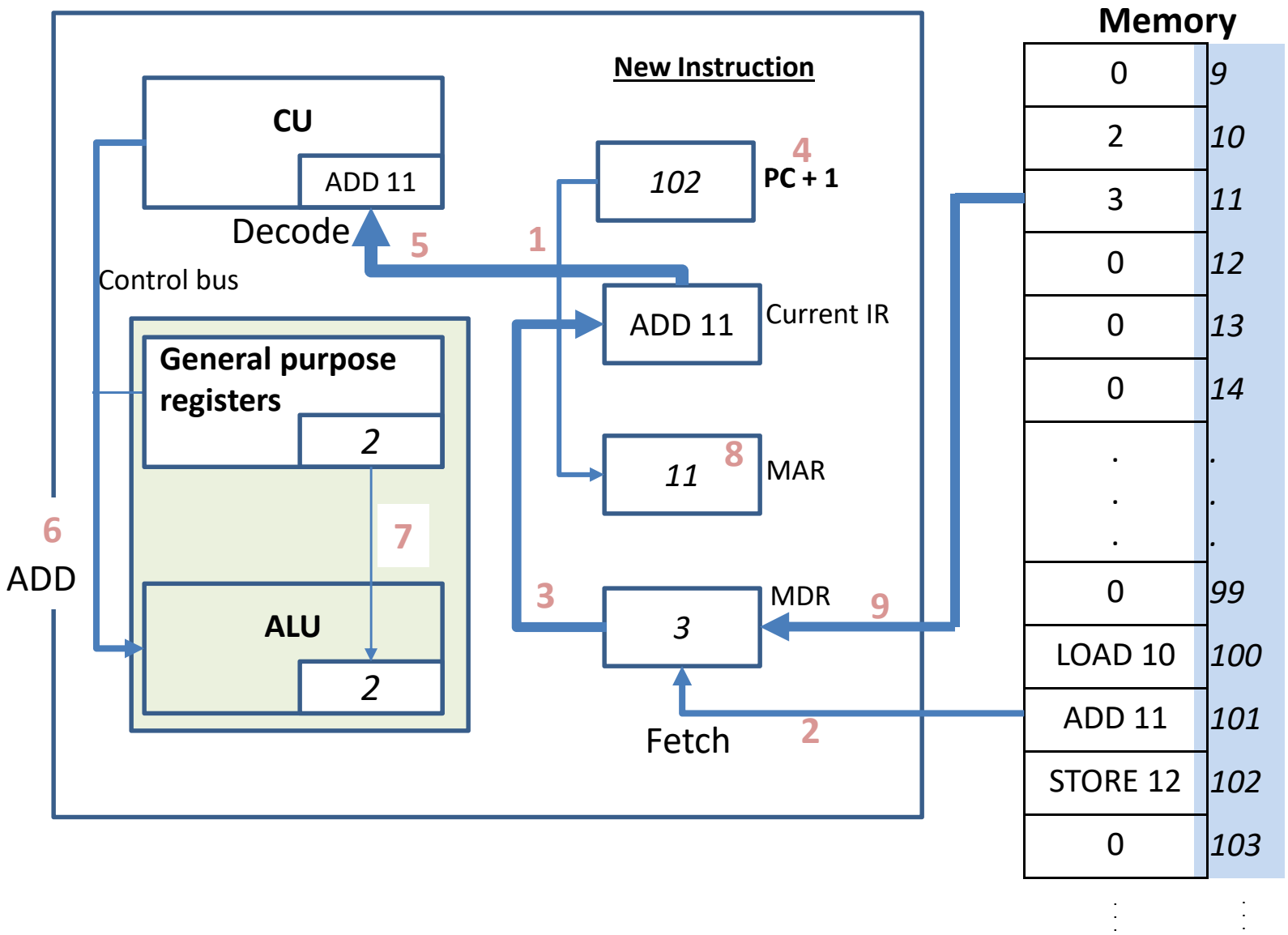
Operation (cont')

A detail example of Fetch-Decode-Execute



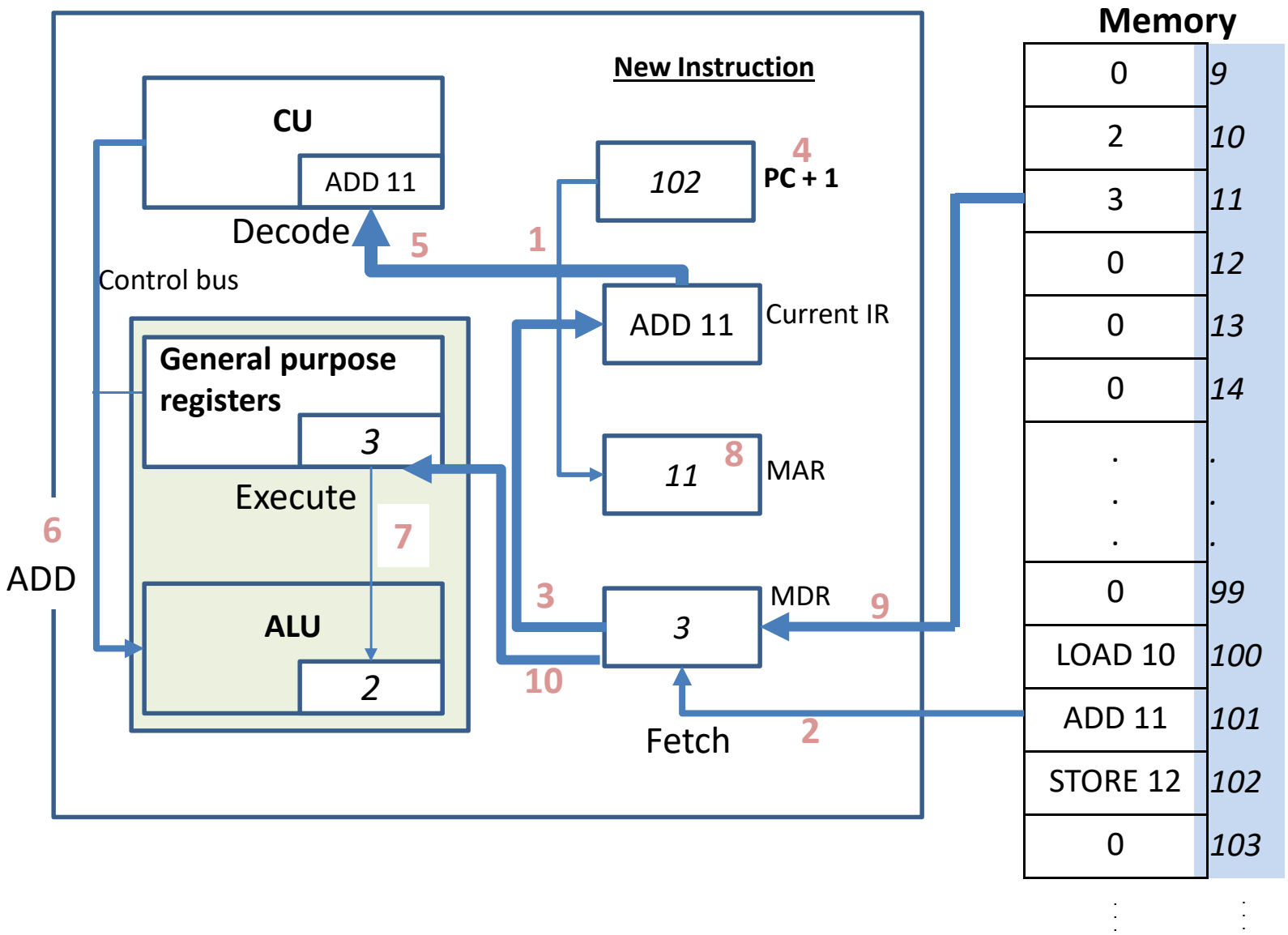
Operation (cont')

A detail example of Fetch-Decode-Execute



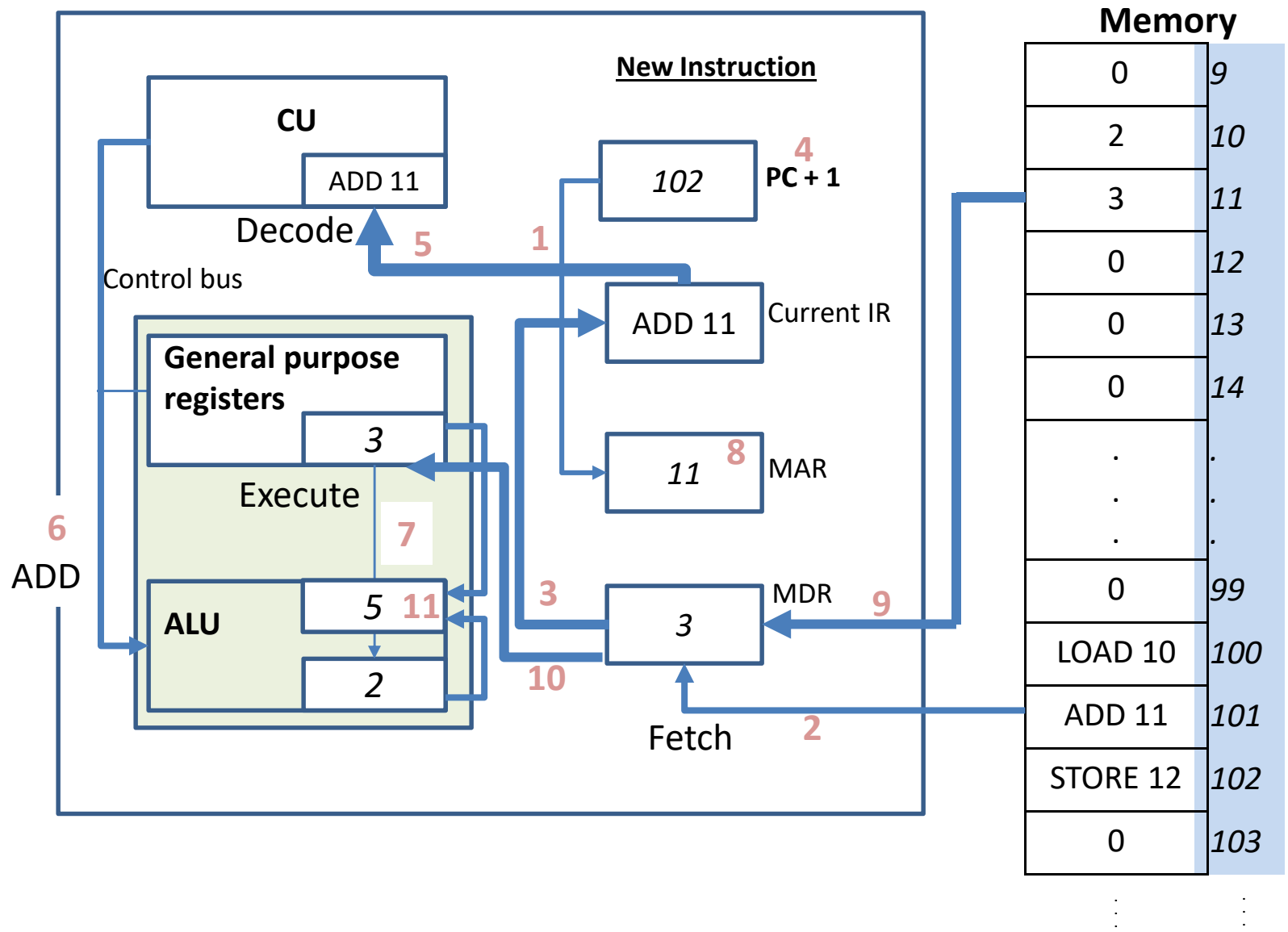
Operation (cont')

A detail example of Fetch-Decode-Execute



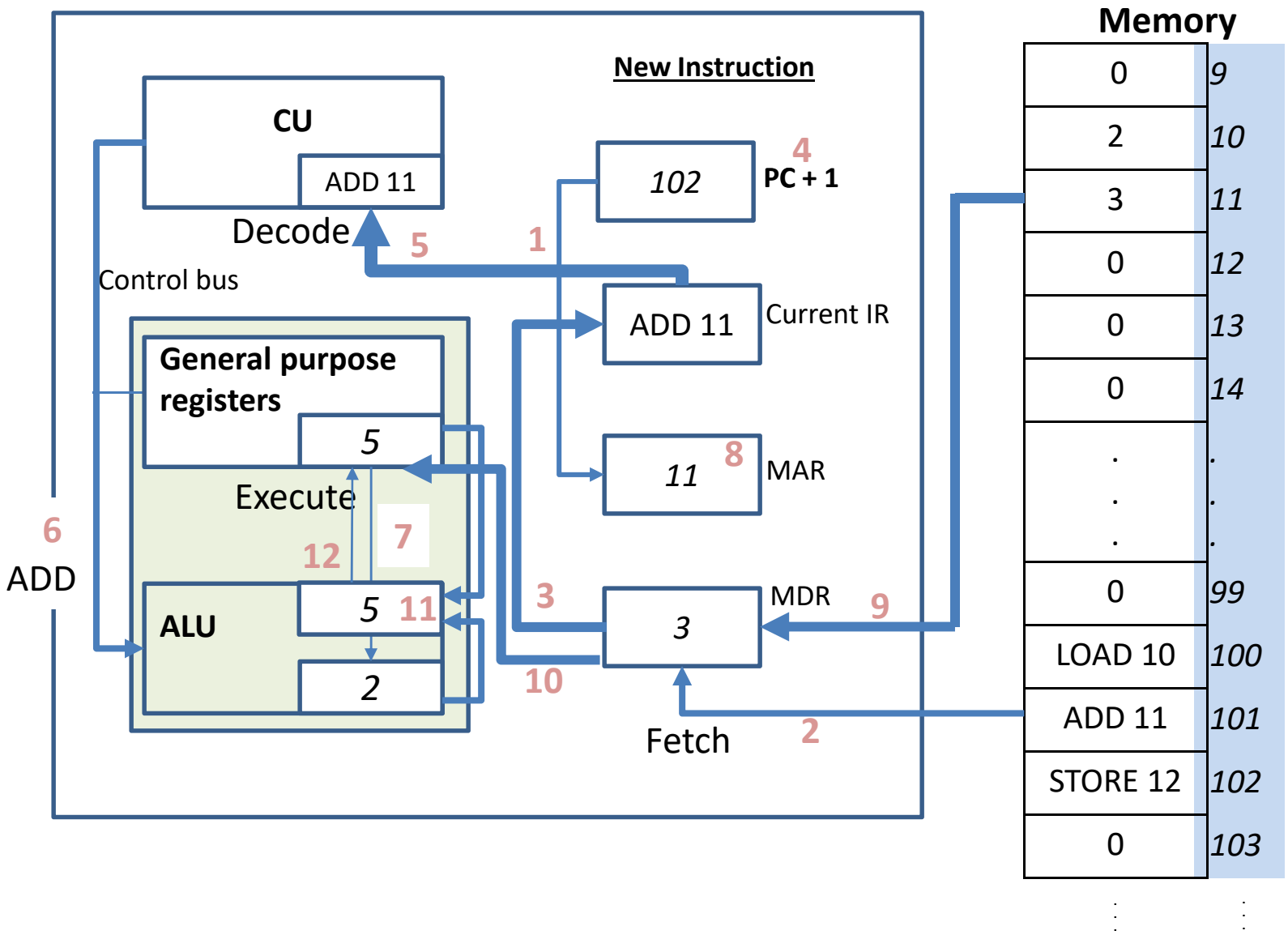
Operation (cont')

A detail example of Fetch-Decode-Execute



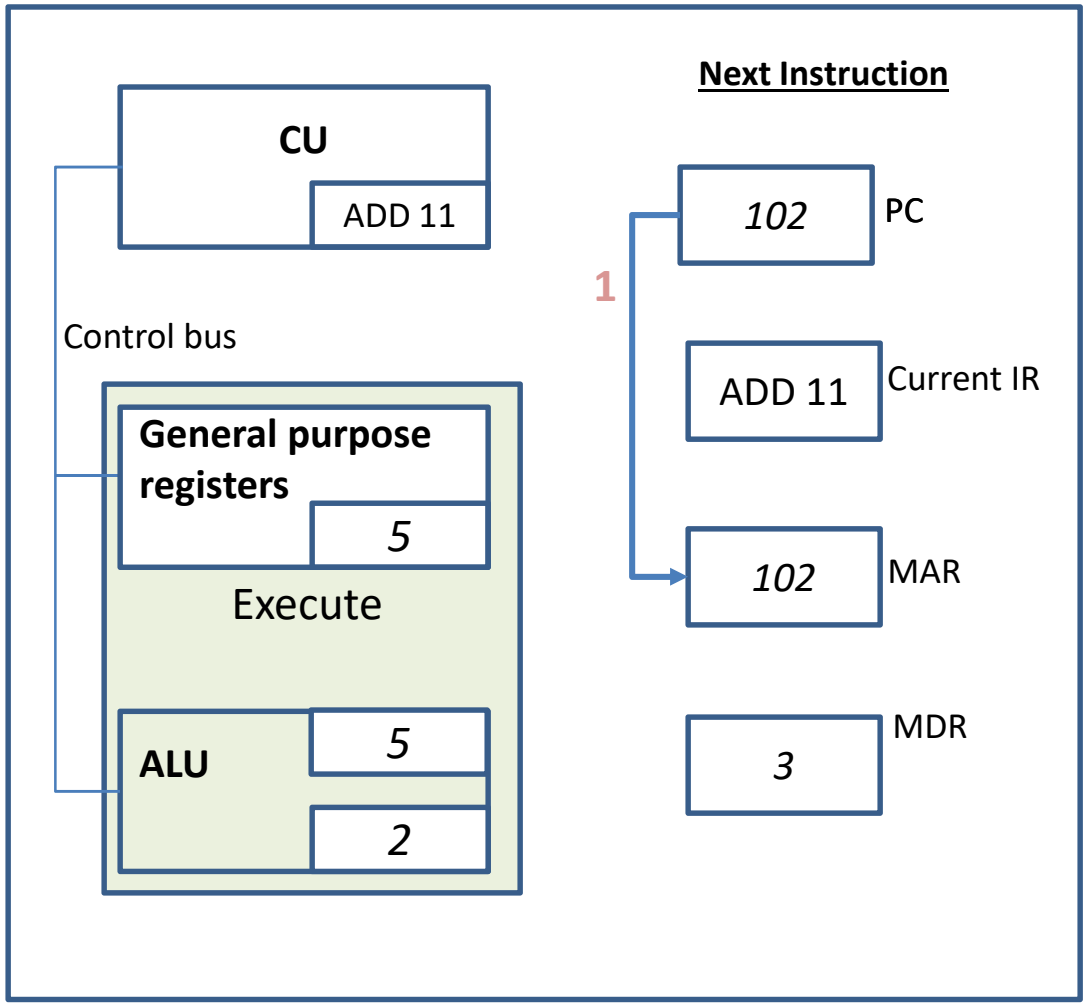
Operation (cont')

A detail example of Fetch-Decode-Execute



Operation (cont')

A detail example of Fetch-Decode-Execute

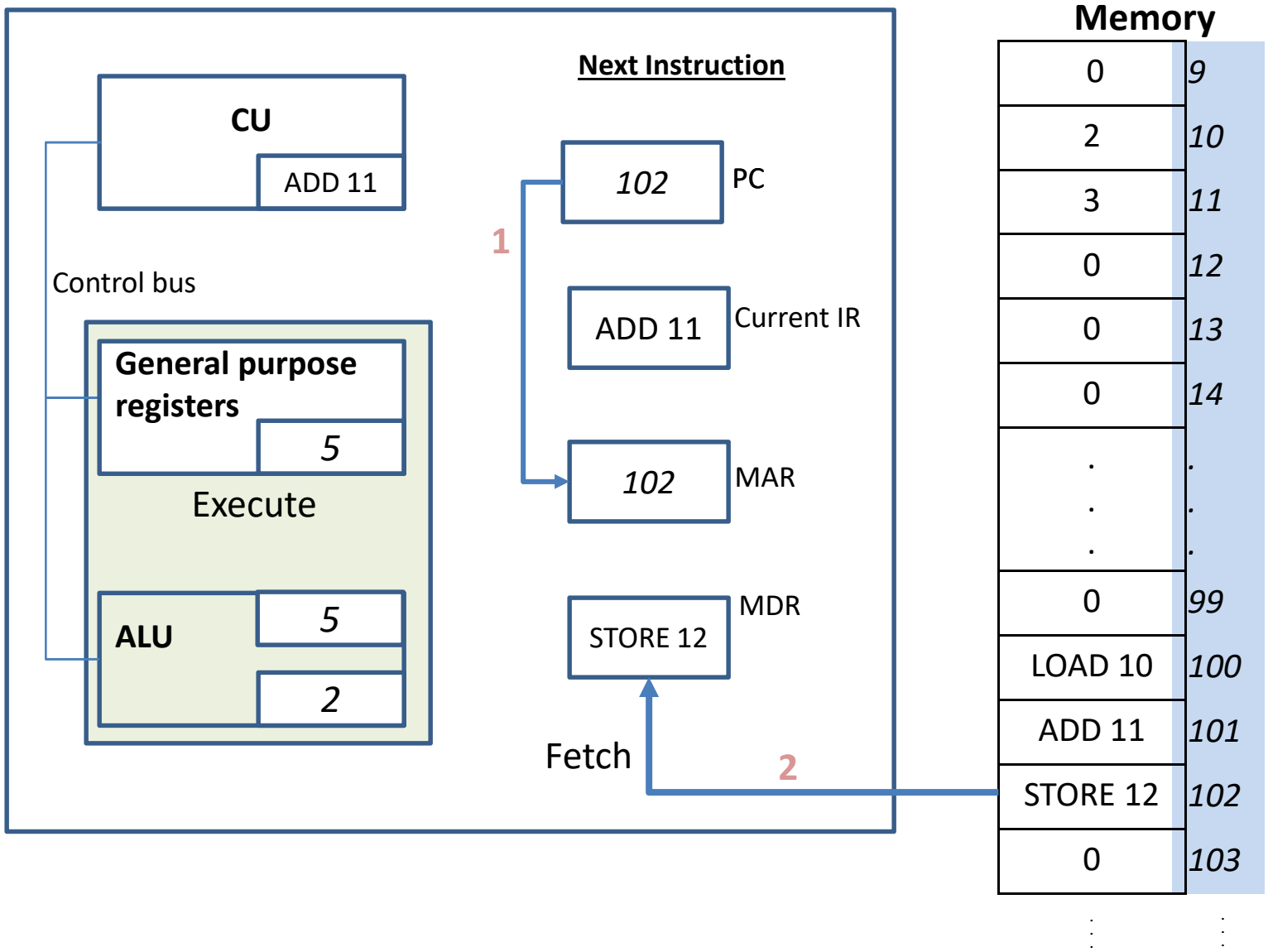


Memory

0	*9*
2	*10*
3	*11*
0	*12*
0	*13*
0	*14*
. . .	. . .
0	*99*
LOAD 10	*100*
ADD 11	*101*
STORE 12	*102*
0	*103*

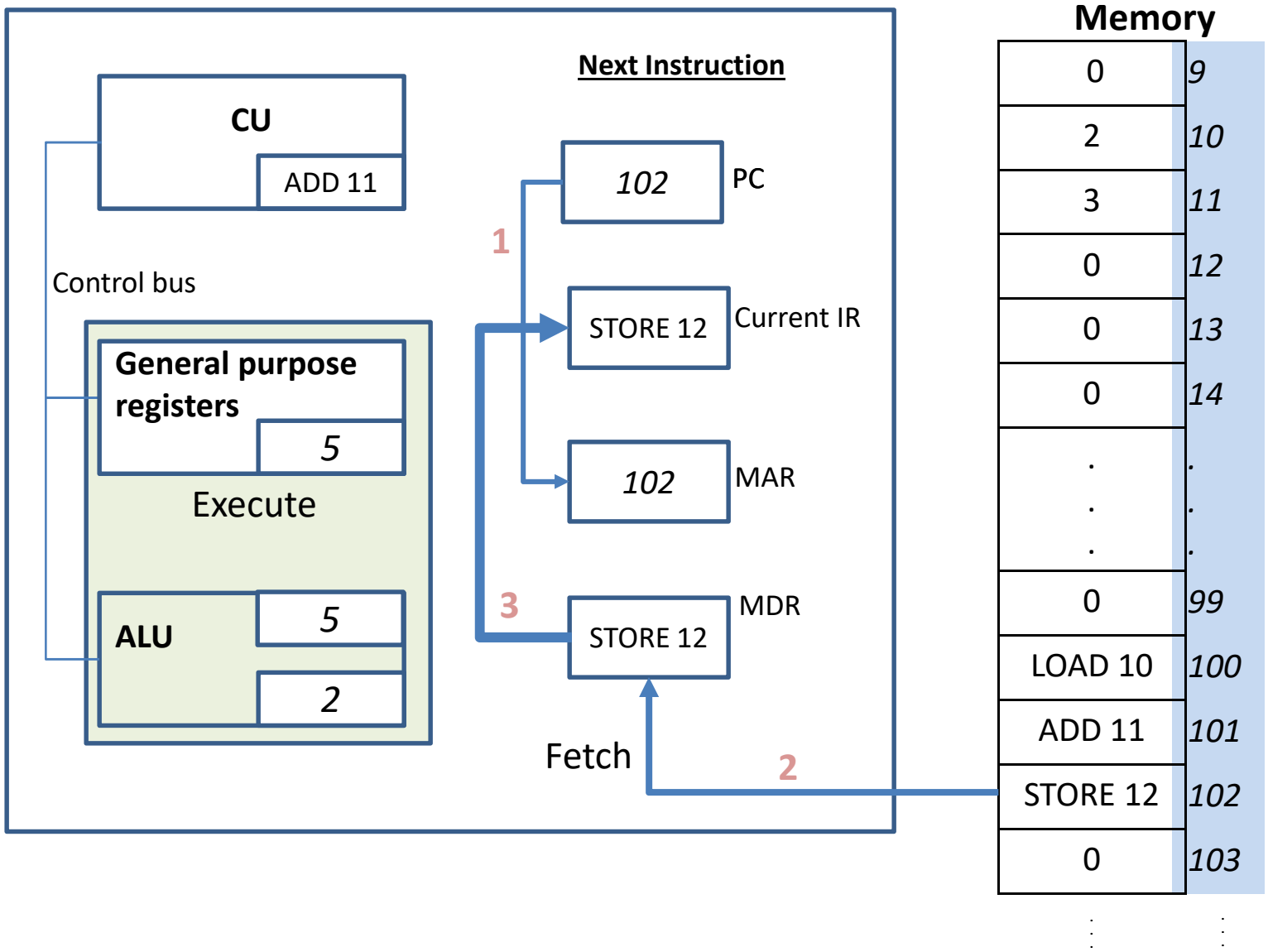
Operation (cont')

A detail example of Fetch-Decode-Execute



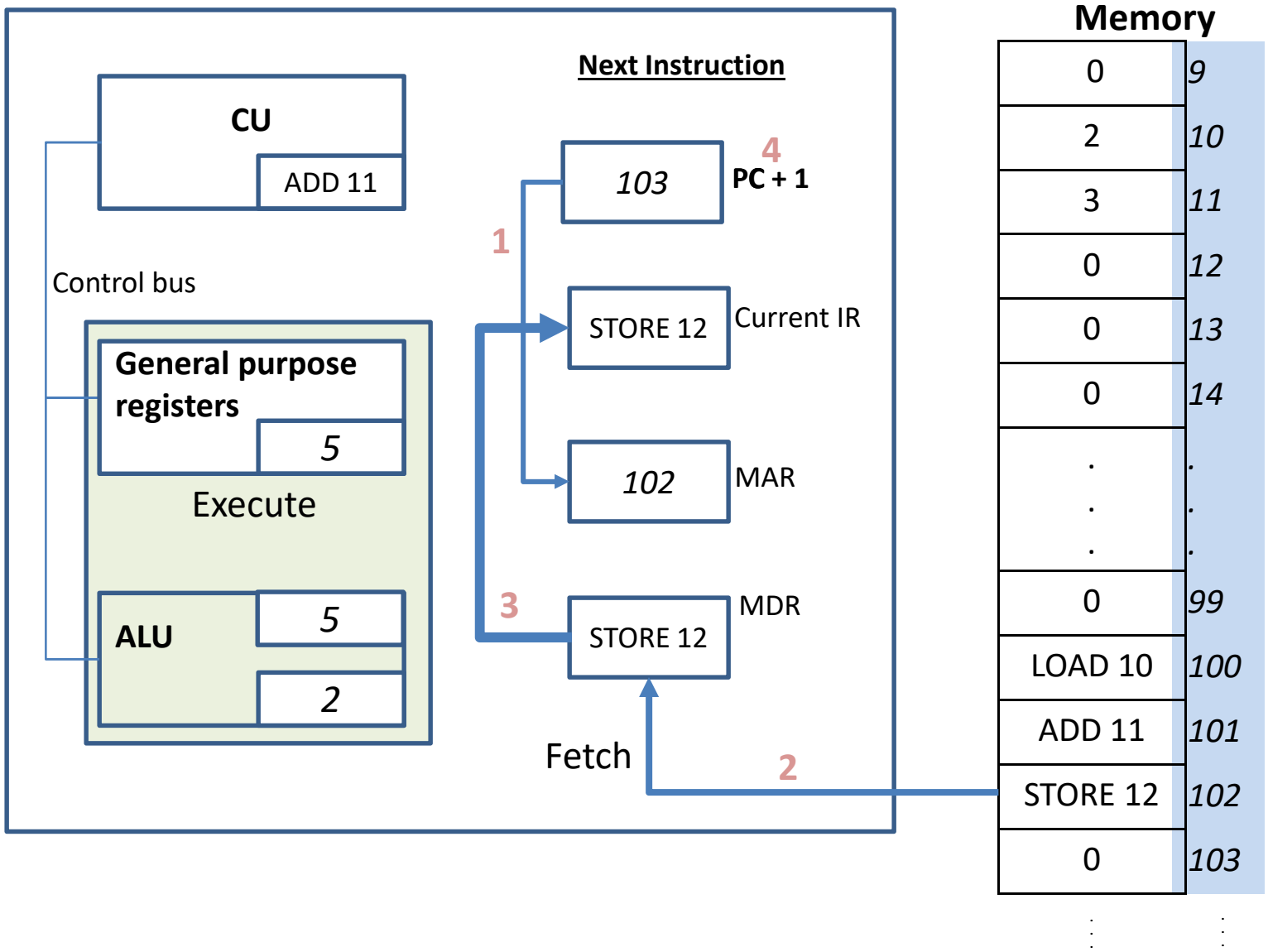
Operation (cont')

A detail example of Fetch-Decode-Execute



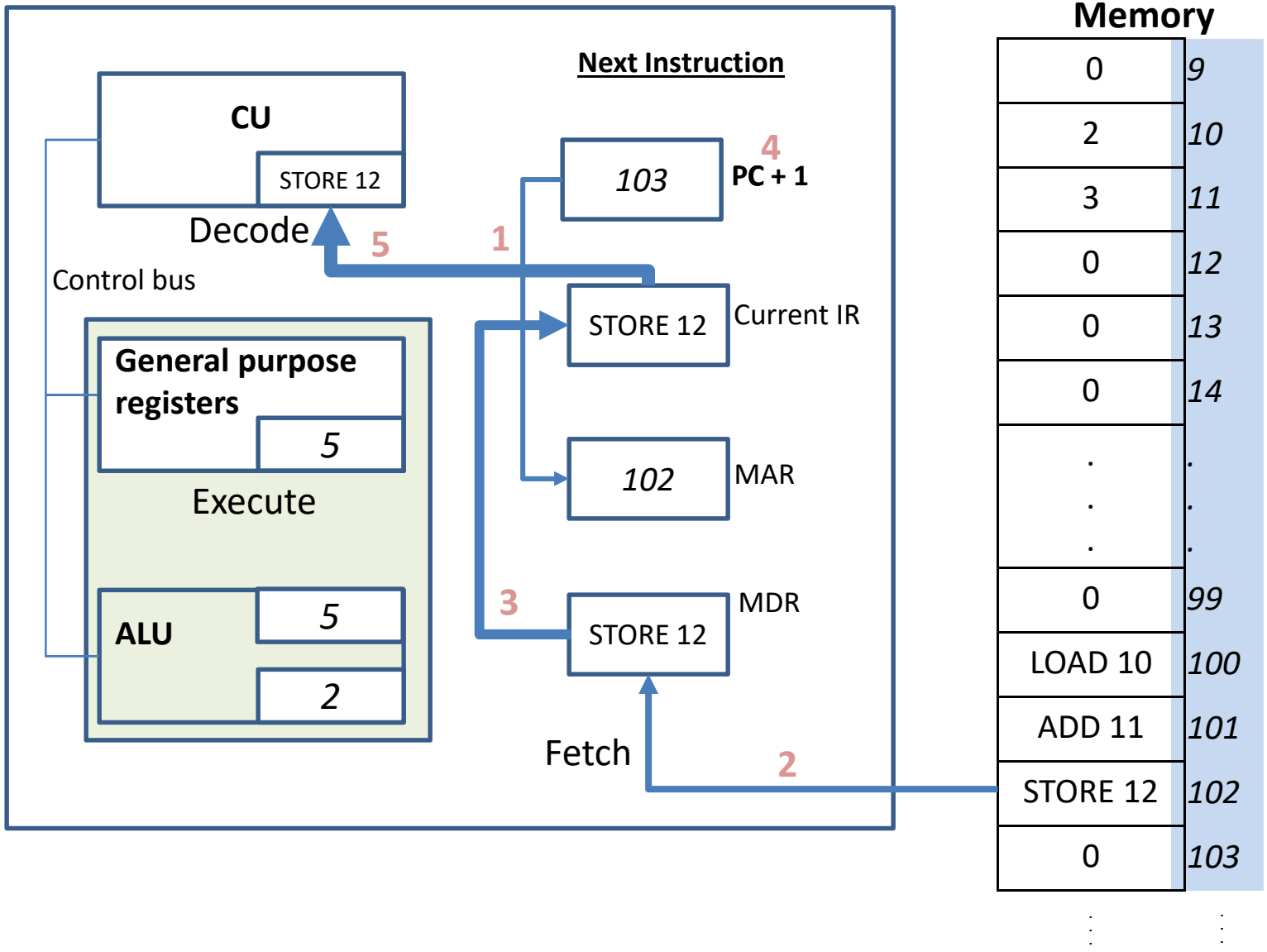
Operation (cont')

A detail example of Fetch-Decode-Execute



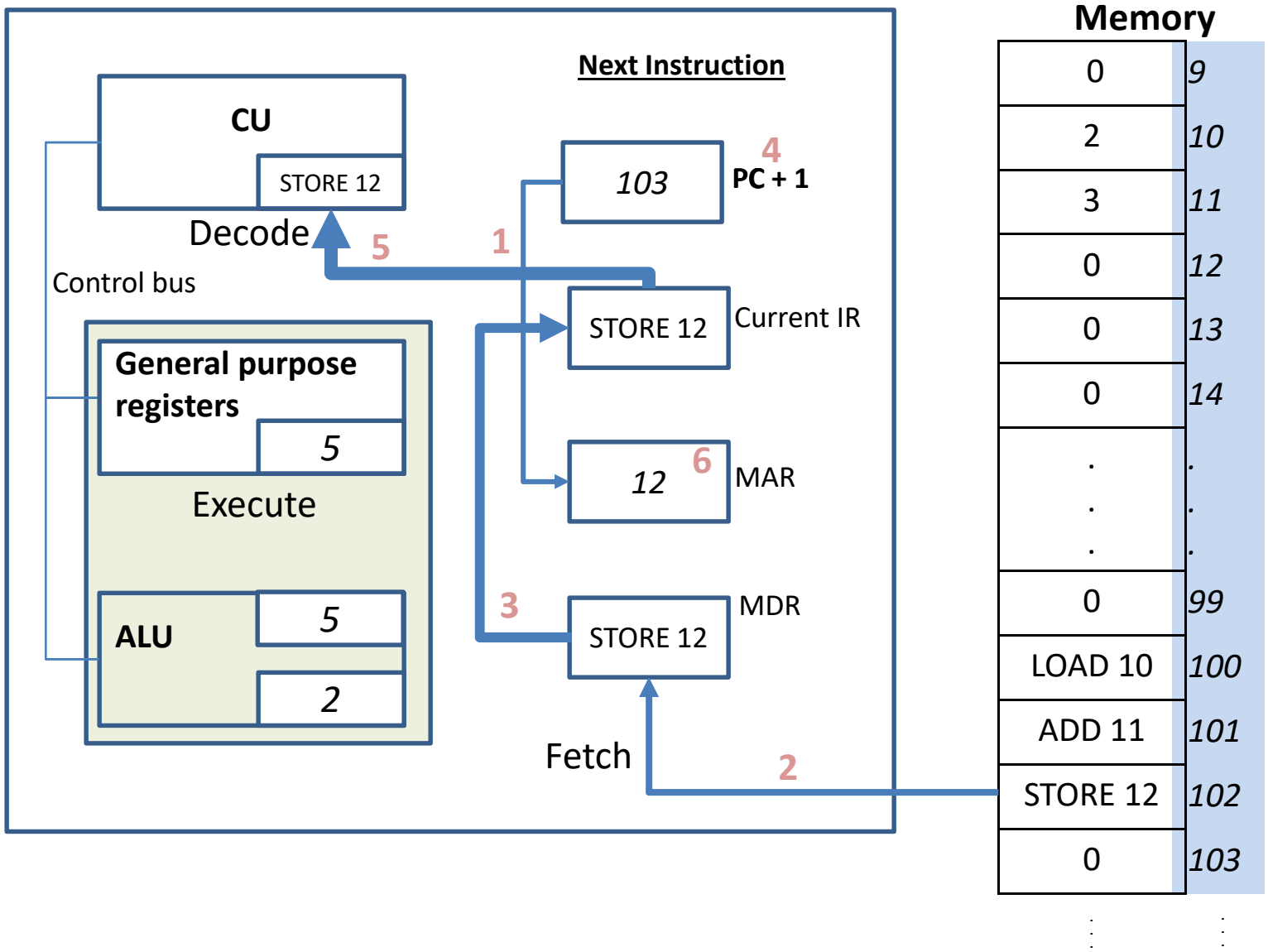
Operation (cont')

A detail example of Fetch-Decode-Execute



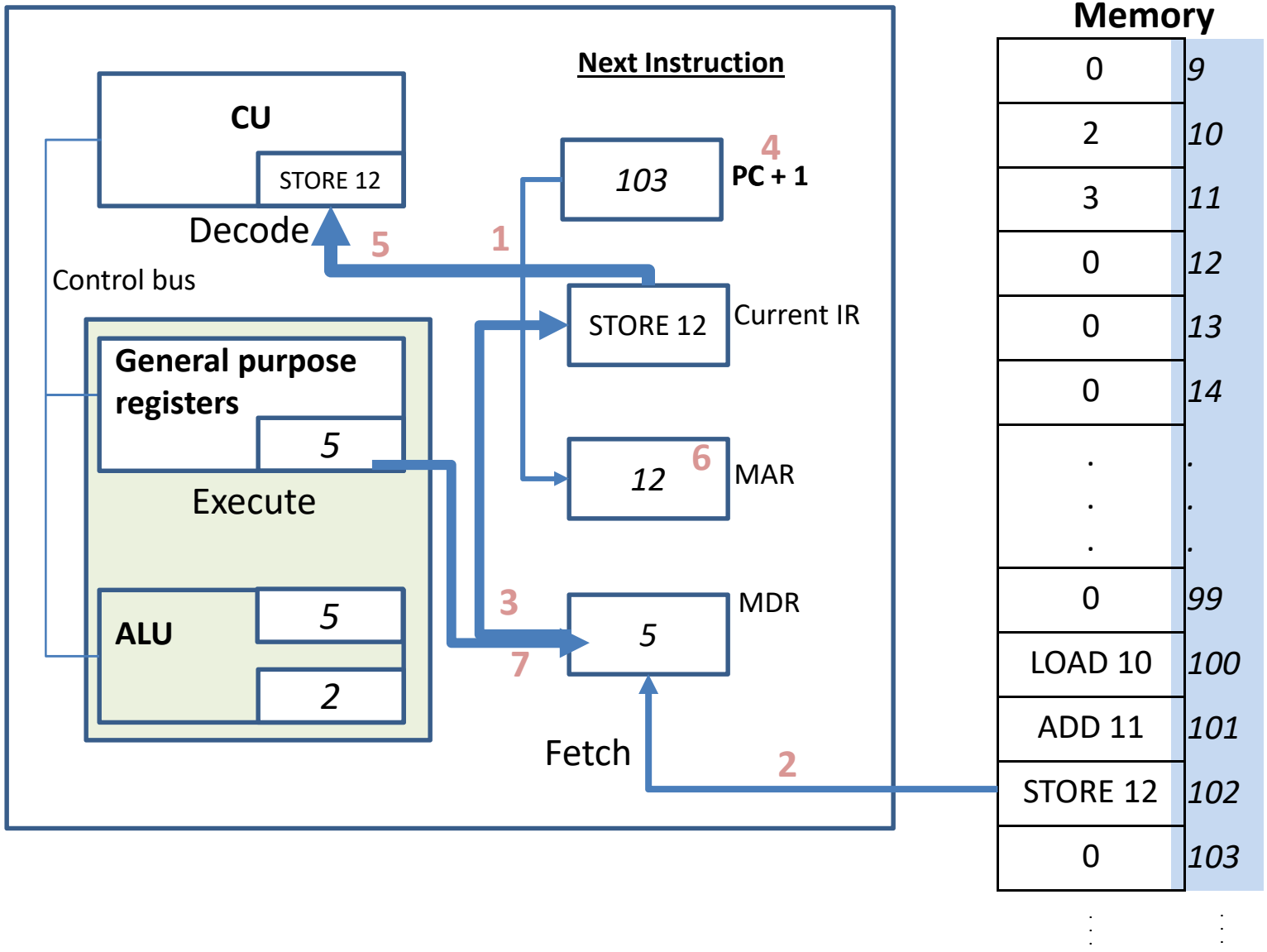
Operation (cont')

A detail example of Fetch-Decode-Execute



Operation (cont')

A detail example of Fetch-Decode-Execute



Operation (cont')

A detail example of Fetch-Decode-Execute