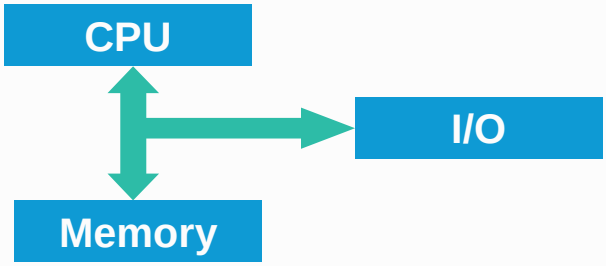
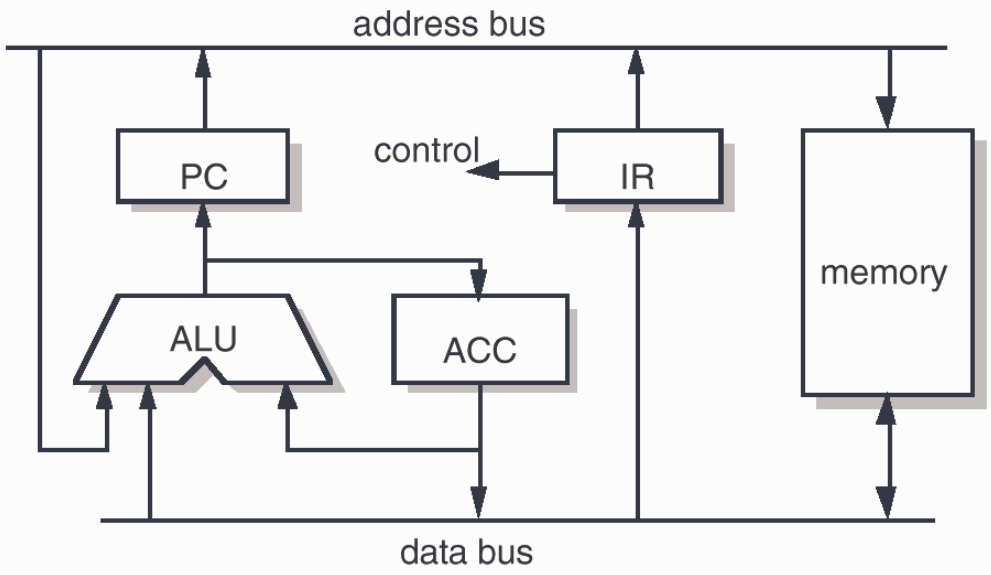
Lecture 2 A Very Simple Processor

- Based on von Neumann model
- Stored program and data in memory
- Central Processing Unit (CPU) contains:
  - Arithmetic/Logic Unit (ALU)
  - Control Unit
  - Registers: fast memory, local to the CPU



- Look into memory and you'll see '1's and '0's. Meaning depends on context: could be program, could be data.
- All computers require at least three types of signals (buses):
  - address bus - which location
  - data bus - carries the contents of the location
  - control bus - governs the information transfer

MU0 - A Very Simple Processor



MU0 - A Very simple processor

- Let us design a simple processor MU0 with 16-bit instruction and data bus and minimal hardware:-
  - Program Counter (PC) - holds address of the next instruction to execute (a register)
  - Accumulator (ACC) - holds data being processed (a register)
  - Instruction Register (IR) - holds current instruction code being executed
  - Arithmetic Logic Unit (ALU) - performs operations on data
- We will only design 8 instructions, but to leave room for expansion, we will allow capacity for 16 instructions
  - so we need 4 bits to identify an instruction: the *opcode*

MU0 - A Very simple processor

- Let us further assume that the memory is word-addressible
  - each 16-bit word has its own location: word 0, word 1, etc.

address	data
0	
1	
…	…

- The 16-bit instruction code (machine code) has a format:

- Note top 4 bits define the operation code (opcode) and the bottom 12 bits define the memory address of the data (the operand)
- This machine can address up to 2^{12} = 4k words = 8k bytes of data

MU0 Instruction Set

Instruction	Opcode	Effect
LDA S	0000	ACC := mem[S]
STO S	0001	mem[S] := ACC
ADD S	0010	ACC := ACC + mem[S]
SUB S	0011	ACC := ACC – mem[S]
JMP S	0100	PC := S
JGE S	0101	if ACC > 0, PC := S
JNE S	0110	if ACC ≠ 0, PC := S
STP	0111	stop

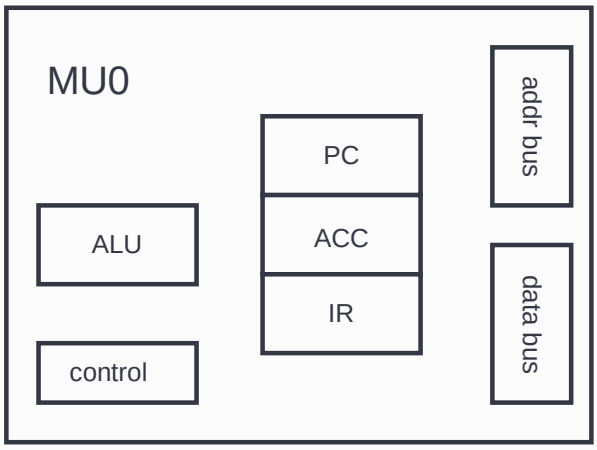
Our First Program

- The simplest use of our microprocessor: add two numbers
  - Let's assume these numbers are stored at two consecutive locations in memory, with addresses 2E and 2F
  - Let's assume we wish to store the result back to memory address 30
- We need to load the accumulator with one value, add the other, and then store the result back into memory

```
LDA 02E          002E
ADD 02F    ---->  202F
STO 030          1030
STP              7???
```

human readable (mnemonic) machine code

Caught in the Act!

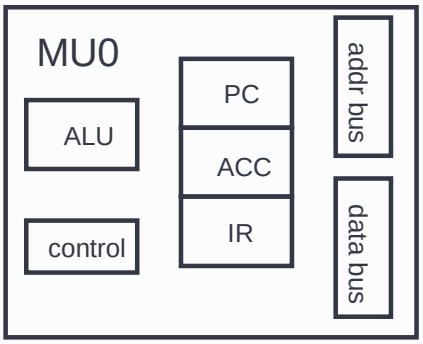


	mnemonic	machine code
000	LDA 02E	**0** 02E
001	ADD 02F	**2** 02F
002	STO 030	**1** 030
003	STP	**7** 000
004	--	--
005	--	--
006	--	--
⋮		
02E	AA	AA
02F	11	11
030	--	--

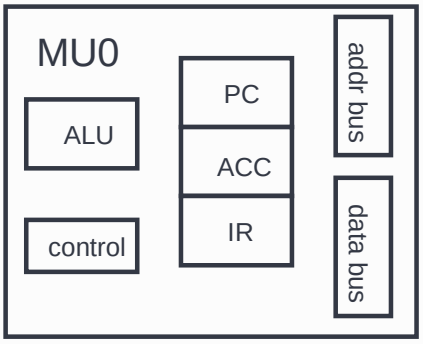
- Initially, we assume PC = 0

Instruction 1: LDA 02E

Cycle 1 (fetch instr and increment PC)

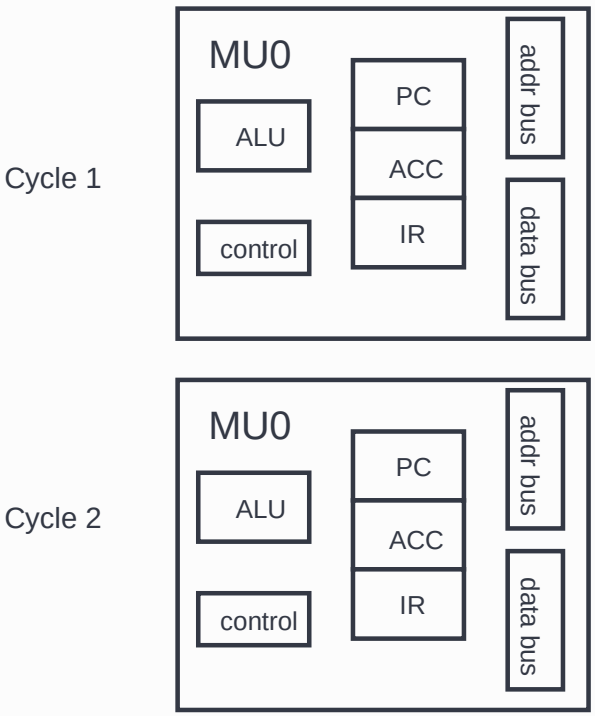


Cycle 2 (execute instruction)



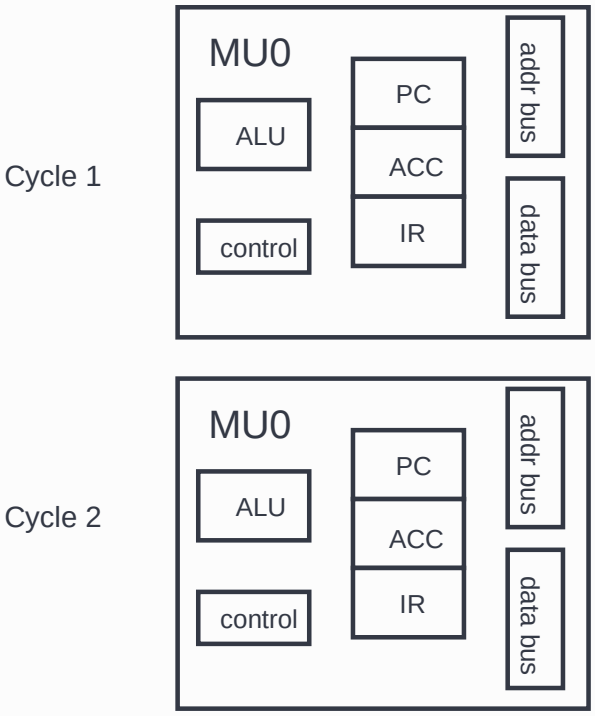
	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	AA
02F	11
030	--

Instruction 2: ADD 02F



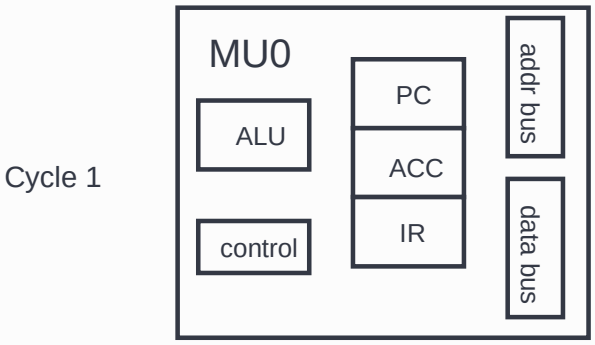
	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	AA
02F	11
030	--

Instruction 3: ST0 030



	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	AA
02F	11
030	BB

Instruction 4: STP



	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	AA
02F	11
030	BB

Key Points

- Microprocessors perform operations depending on instruction codes stored in memory
- Instructions usually have two parts:
  - Opcode - determines what is to be done
  - Operand - specifies where/what is the data
- Program Counter (PC) - address of instruction
- PC incremented automatically each time it is used
- The number of clock cycles taken by a MU0 instruction is the same as the number of memory access it makes.
  - LDA, STO, ADD, SUB therefore take 2 clock cycles each: one to fetch (and decode) the instruction, a second to fetch (and operate on) the data
  - JMP, JGE, JNE, STP only need one memory read and therefore can be executed in one clock cycle.

Key Points (con't)

- Memory contains both programs and data
- Program area and data area in memory are usually well separated
- ALU is responsible for arithmetic and logic functions
- There are usually one or more general purpose registers for storing results or memory addresses (MU0 only has ACC)
- Fetching data from inside the CPU is much faster than from external memory
- Assume MU0 will always reset to start execution from address 000_{16}.