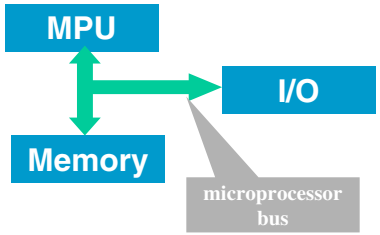
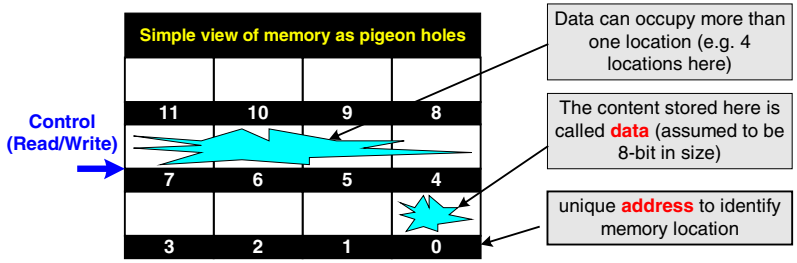
Lecture 3 A Very Simple Processor

- Based on von Neumann model
- Instruction for the microprocessor (stored program) and the data for computation are both stored in **memory**
- **Microprocessing Unit** (MPU) contains:
  - Arithmetic/Logic Unit (ALU)
  - Control Unit
  - Registers



- **Input/output** block interfaces to the outside world (e.g. user, disc storage etc.)
- The communication between memory, MPU and I/O is through the **microprocessor bus**

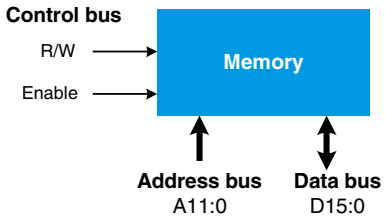
Memory



- Most microprocessors uses one unique address for each byte of storage.
- Multiple locations can be concatenated to form a larger data word (for example, 4 locations to form one 32-bit word).
- In this case, the memory word is identified by the lowest byte address (in this case, addresses 0, 4 ,8 …. etc.).

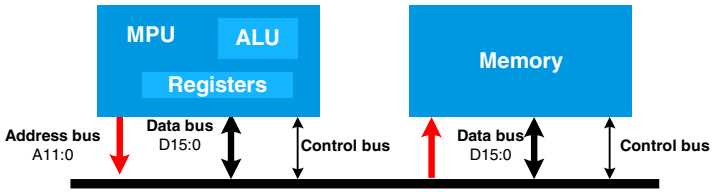
Memory

- Computers communicate with memory using 3 types of signals (**buses**):
  - **address bus** - determines the location of memory
  - **data bus** - carries the contents of the location
  - **control bus** - governs the information transfer



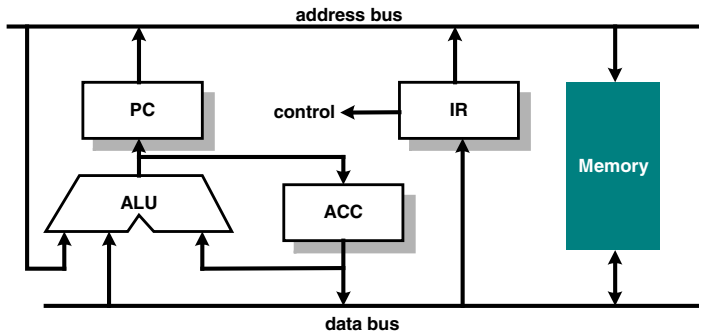
- Look into memory, see '1's and '0's. Meaning depends on **context**.
- The width of the address bus determines the size of memory (i.e. how many location).
- The width of the data bus determines the size of content (i.e. how many bits can each location store).

Memory, registers and ALU



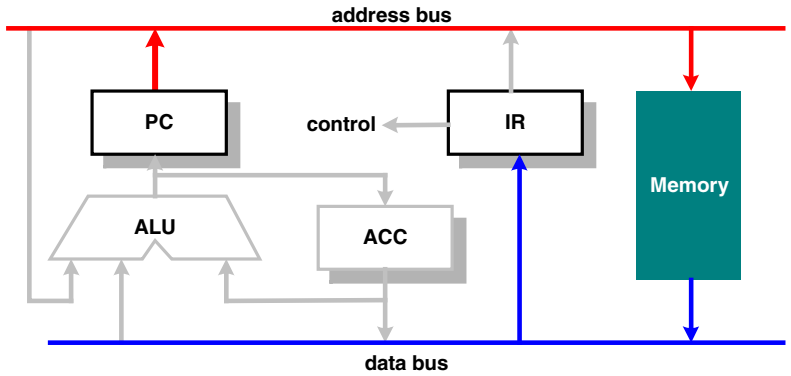
- A microprossor (MPU) must contain:
  - An Arithmetic and Logic Unit (**ALU**) to perform computation
  - Some internal storages called **Registers** to store temporary data and instructions
- To run a program, the MPU must:
  - 1) **Fetch instruction** - Supply instruction address and read an instruction from memory on the data bus
  - 2) **Decode instruction** - Stored instruction is interpreted by MPU
  - 3) **Fetch operand** - Supply address of data and read data into MPU
  - 4) **Execution instruction** - Perform the necessary action by MPU

MU0 - A Very Simple Processor



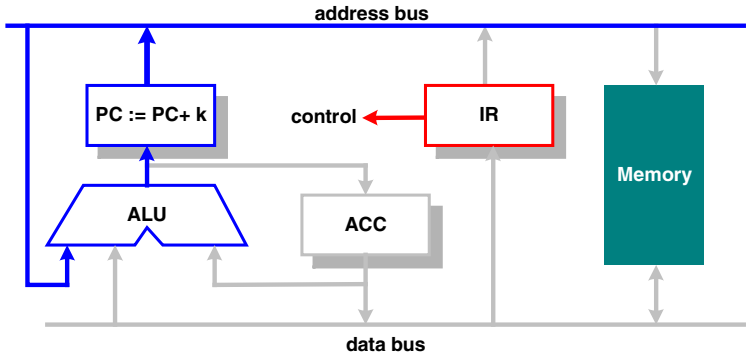
- Let us design a simple processor MU0 with 16-bit instruction and minimal hardware:-
  - Program Counter (PC) - holds address of the next instruction to execute
  - Accumulator (ACC) - holds data being processed
  - Arithmetic Logic Unit (ALU) - performs operations on data
  - Instruction Register (IR) - holds current instruction code being executed

Instruction execution step 1: Instruction Fetch



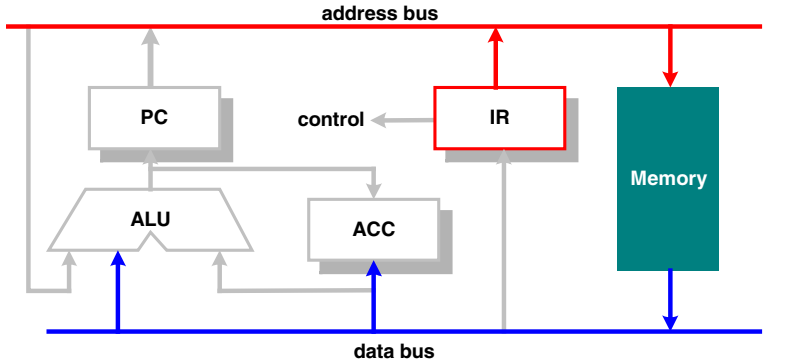
- MPU outputs value of program counter (PC) on address bus.
- Memory puts contents at the instruction address on the data bus.
- Instruction is stored in instruction registers (IR).

Step 2: Instruction Decode



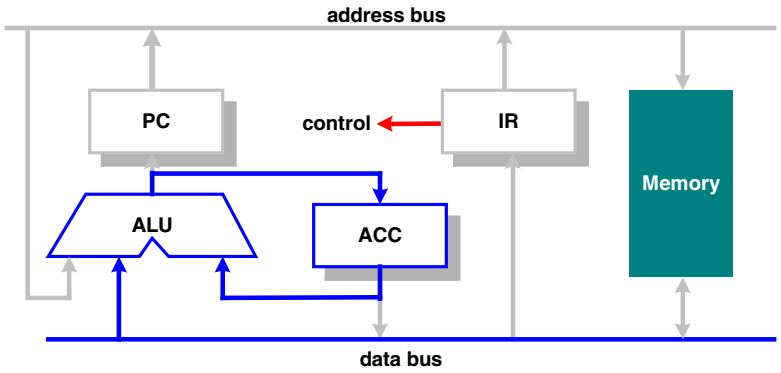
- The instruction word stored in IR is decoded by internal logic to provide control signals to ALU and other internal circuits inside MPU.
- Program counter value (PC) is pushed onto the address bus, the ALU increment this value by **k** and put it back into the Program Counter.

Step 3: Operand Fetch



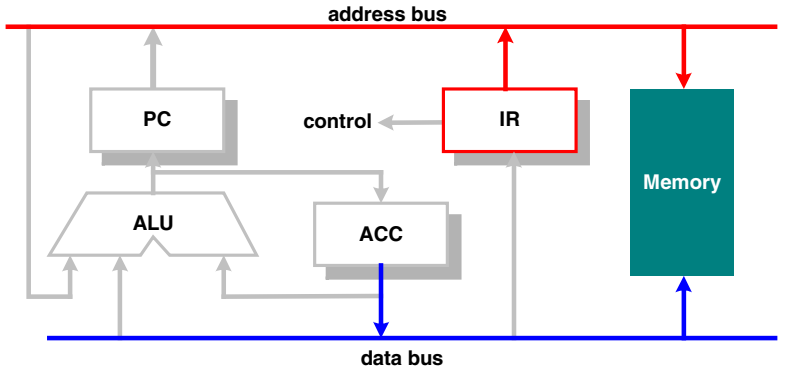
- The instruction register provides the address of the data to be processed (i.e. **operand address**).
- Memory supplies the operand data on the data bus to the MPU, ready for processing either by the ALU or the ACC.

Step 4: Execute instruction



- Processing is performed on the operand by the ALU according to the instruction.
- The result is put back into the Accumulator (ACC).

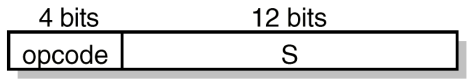
Step 5: Write-back (may not exist)



- This step may not be needed.
- The result from the Accumulator is written back into memory. In many processors, this will be done as a separate instruction.

MU0 instructions

- Let us further assume that the processor only has 8 instructions and can only access a maximum of 8k byte (2^{12}) of memory. This implies that the address bus is only 12-bit wide.
- We also assume that this is a 16-bit MPU and **ALL** instructions are 16 bits wide.
- The 16-bit instruction code (machine code) has a format:



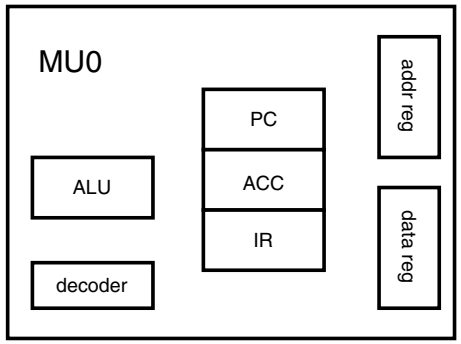
- Note that top 4 bits define the operation code (opcode), i.e. it defines what operation the instruction is to perform.
- The bottom 12 bits define the memory address of the operand data.

MU0 Instruction Set

Instruction	Opcode	Effect
LDA S	0000	ACC := mem_{16}[S]
STO S	0001	mem_{16}[S]:= ACC
ADD S	0010	ACC := ACC + mem_{16}[S]
SUB S	0011	ACC := ACC - mem_{16}[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

- 2 load/store instructions: LDA, STO
- 2 computation instructions: ADD, SUB
- 4 control flow instructions: JMP, JGE, JNE, STP

Caught in the Act!

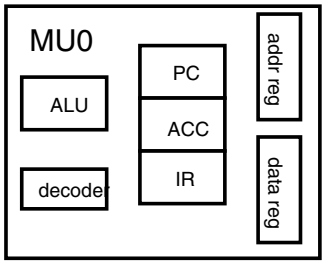


	nmemonic	machine code
000	LDA 02E	**0** 02E
001	ADD 02F	**2** 02F
002	STO 030	**1** 030
003	STP	**7** 000
004	--	--
005	--	--
006	--	--
⋮		
02E	ABCD	ABCD
02F	4321	4321
030	--	--

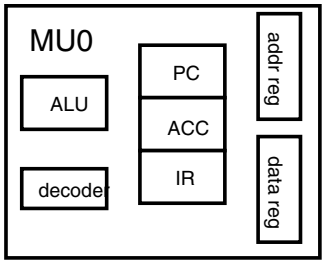
- CPU reading the first op-code (the following slides are to be completed during lecture).

Instruction 1: LDA 02E

Cycle 1



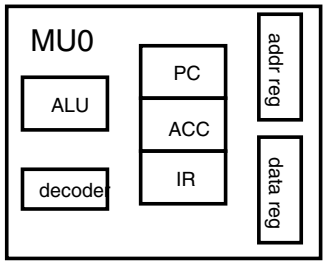
Cycle 2



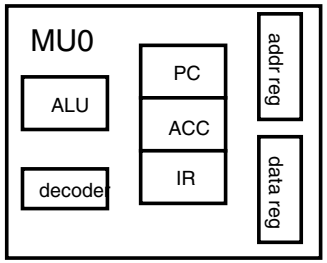
	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	**ABCD**
02F	4321
030	--

Instruction 2: ADD 02F

Cycle 1



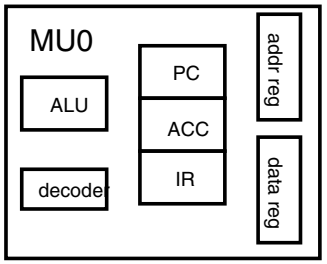
Cycle 2



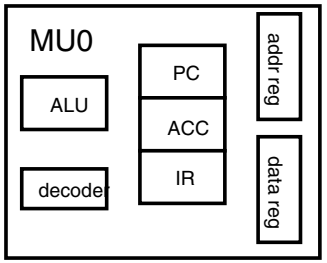
	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	ABCD
02F	**4321**
030	--

Instruction 3: ST0 030

Cycle 1

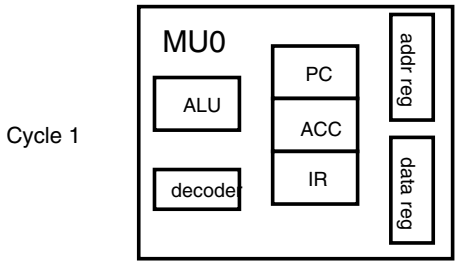


Cycle 2



	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	ABCD
02F	4321
030	**EEEE**

Instruction 4: STP



Cycle 1

	machine code
000	**0** 02E
001	**2** 02F
002	**1** 030
003	**7** 000
004	--
005	--
006	--
⋮	
02E	ABCD
02F	4321
030	EEEE

Operation of the processor

- The operation of most processors are governed by a clock signal.
- For MU0, we assume that:
  - The number of clock cycles taken by an instruction is the same as the number of memory access it makes.
  - LDA, STO, ADD, SUB therefore takes 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.
  - Program Counter (PC) - its content is incremented every time it is used (i.e. it also points to the next instruction).
  - No PC incrementer circuit is needed for MU0.
  - The processor must start from a **known state**. Therefore, there is always a reset signal to initialise the processor on power-up.
  - Assume MU0 will always reset to start execution from address 000_{16}.

Summary of key points

- Microprocessors performs operations depending on instruction codes stored in memory.
- Instruction usually has two parts:
  - Opcode - determines what is to be done
  - Operand - specifies where/what is the data
- Program Counter (PC) - address of current instruction code
- PC incremented automatically each time it is used.
- The number of clock cycles taken by an instruction is the same as the number of memory access it makes.
  - LDA, STO, ADD, SUB therefore takes 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 program and data. A peek into memory will tell you very little except a bunch of '1's and '0's.
- Program area and data area in memory are usually well separated.
- ALU is responsible for arithmetic and logic functions.
- There is always at least one register known as **accumulator** where the result from ALU is stored.
- There is usually one or more **general purpose register** for storing results or memory addresses.
- Fetching data from inside the CPU is much faster than from external memory.
- The processor must start from a **known state**. Therefore, there is always a reset signal to initialise the processor on power-up.
- Assume MU0 will always reset to start execution from address 000_{16}.