The two cases of the Fetch and Execute Cycle:

A. What is fetched is actually an address:

1. fetch (reference)

2. decode (and realize that what was fetched was an address)

2.1 fetch the data at the address which was just fetched

2.2 decode (this will now be data/instructions, rather than an address)

3. execute

4. store partial result (in CPU accumulator register)

(perform next fetch)

5. (eventually) store result (in RAM)

B. When what is fetched is data or instructions

1. fetch (data)

2. decode

3. execute

4. store partial result (in CPU accumulator register)

(perform next fetch)

5. (eventually) store result (in RAM)

Important Registers Used in the Fetch & Execute Cycle:

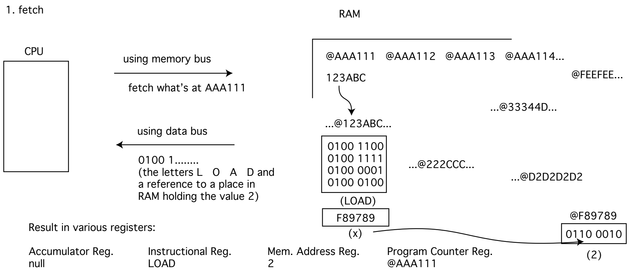
Accumulator Register - temporarily stores a value that is in the process of being calculated, one step at a time.
Instruction Register - holds the instructions, one at a time.
Memory Address Register - holds either:
- the memory address of the data ("operand") to be used by the instruction
- the memory address to which result to be written ("stored")

Program Counter Register - holds the next memory address in line to be fetched.

Example of 2 + 3 =

Step 1; in assember code: LOAD x

1. fetch



Result in various registers:

Accumulator Reg.	Instructional Reg.	Mem. Address Reg.	Program Counter Reg.
null	LOAD	2	@AAA111

2. Decode.

3. Execute.

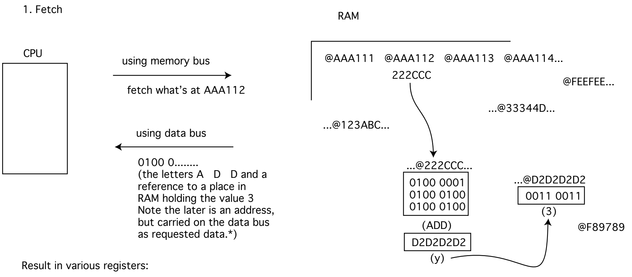
4. Store in Accumulator.

Result in various registers:

Accumulator Reg.	Instructional Reg.	Mem. Address Reg.	Program Counter Reg.
2	null	null	@AAA112

Step 2; in assembler code: ADD y

1. Fetch



Result in various registers:

Accumulator Reg.	Instructional Reg.	Mem. Address Reg.	Program Counter Reg.
2	ADD	3	@AAA112

2. Decode.

3. Execute.

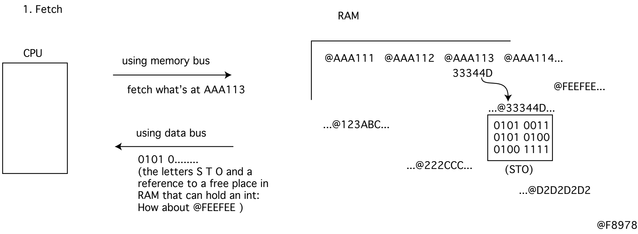
4. Store in Accumulator:

Result in various registers:

Accumulator Reg.	Instructional Reg.	Mem. Address Reg.	Program Counter Reg.
5	null	null	@AAA113

Step 3: in assembler code: STO

1. Fetch



Result in various registers:

Accumulator Reg.	Instructional Reg.	Mem. Address Reg.	Program Counter Reg.
5	STO	@FEEFEE...	@AAA114

2. Decode.

3. Execute.

4. Store in Accumulator - Nope, not this time: But Stored back to RAM (not shown) .

Result in various registers:

Accumulator Reg.	Instructional Reg.	Mem. Address Reg.	Program Counter Reg.
null	null	null	@AAA114

Address Bus Versus Data Bus

Basically the address bus carries requests from the CPU to the RAM for information stored at various specific memory addresses. It is (in fact will have to be) the physical width of the system architecture - 32 wires wide, or 64 wires wide - so as to accommodate all the combinations of addresses being sent at one time.

Meantime, regarding the data bus, most of the time it finds itself carrying back data from the RAM; that data will be either in the form of instructions, or it will be data itself. And for objects, that "data" will be references. But it's still data, carried on the data bus, even though the data is an address. And the data bus is two-way, since at times it will carry processed data back from the accumulator register to be stored in RAM.

And remember there are two buses, internal and external. The internal bus is made up of both the address and the data bus, and allows for transferral of addresses and data between the CPU and the cache and RAM memory.