Chapter 7
The Basic Operations of a Microprocessor

7.1 Instructions and Data

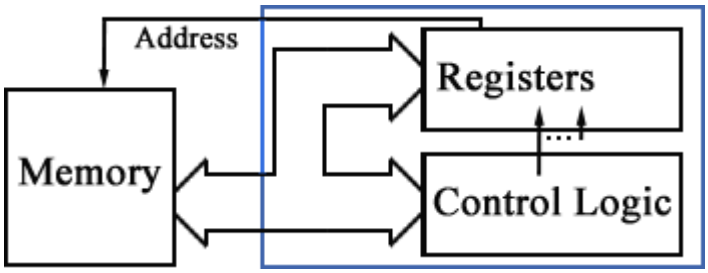


Figure 7.1: Simpilified Microprocessor Design Diagram

A simplified diagram of the microprocessor is given above. The binary information from the memory is applied to both the registers and the control circuitry, and is interpreted in different ways:

1. The binary pattern applied to the control logic is the instruction, and will cause the control logic to generate the necessary sequence of **micro-operations** that will accomplish the desired **macro-operation** indicated by the instruction.

2. The data information from the memory is transferred to and from the registers within the CPU, under control of the micro-operations generated by the control logic.

NOTE:

Both instructions and data share a common path to and from the memory, although they are routed to different places within the CPU. This common data path is the '**data bus**'. Since both the instructions and the data are accessed from the memory in exactly the same way, there is thus no physical difference between them; the operation of the CPU is what determines whether a given word in memory contains an instruction or a data byte.

For an 8 bit CPU, the data bus width (which is the same as the memory width) is 8 bits, which implies that:

1. **Instructions** are 8 bits wide, and there are thus 2^8 or 256 different combinations, or 256 possible instructions. Not all of these will be used; often only a smaller subset will be valid instructions. The remainder are called 'illegal instructions', and if the CPU attempts to treat them as instructions, the results will certainly be unpredictable and will often be disastrous.

2. **Data** which are transferred to and from the registers is in units of 8 bits. Thus if a register is 16 bits wide, its contents will require two bytes of memory to store.

7.2 The Control Logic

The control circuitry is in itself a small sequential circuit. Its function is to generate the control signals for the micro-operations to transfer and manipulate the data in the registers and ALU of the microprocessor. The inputs to the control circuitry are the bits of the instruction to be executed; the bits of each particular instruction will thus cause the control circuitry to cycle through a unique state sequence in order to generate the correct micro-operations to perform the instruction.

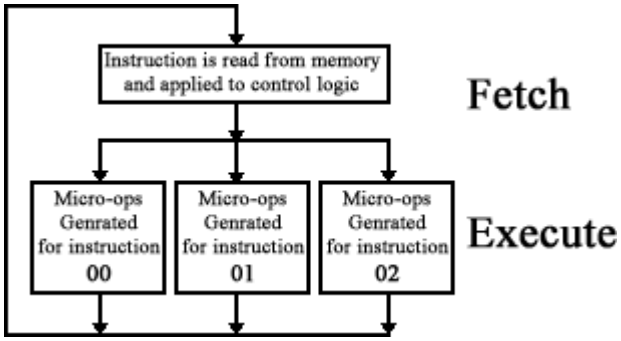


Figure 7.2: The Control Logic Sequence

This sequence is shown in a very broad form here:

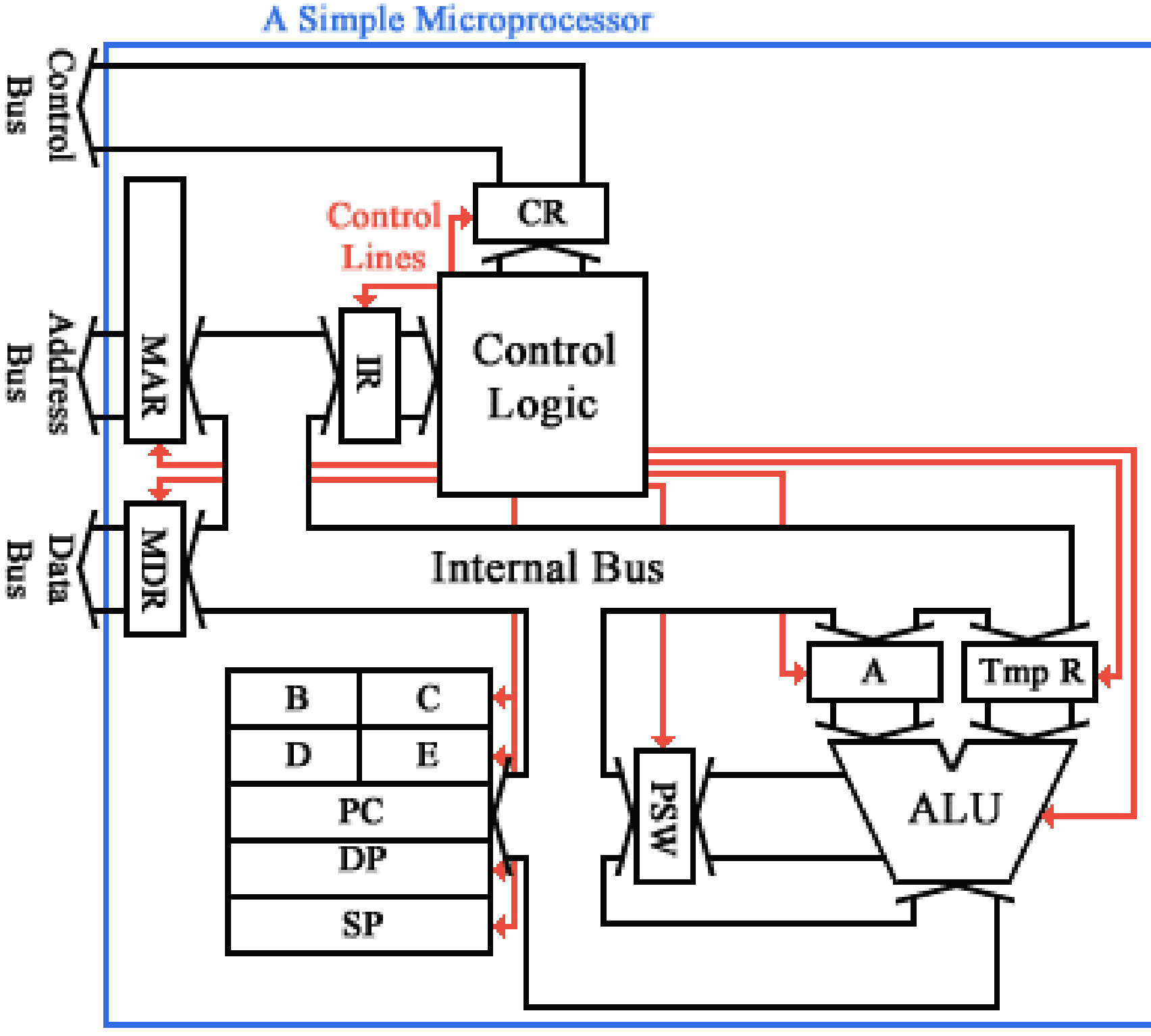
During the first part of the cycle, the instruction is read from memory and applied to the inputs of the control logic. Note that the micro-operations to achieve this are generated by the control logic itself; this first part of the cycle is always performed, and the control logic always generates the same micro-operations. It does not need to have any instruction applied to the inputs to do this. This first stage of the instruction processing is called the **fetch cycle**.

The second part of the instruction processing is called the **execute cycle**. The operations of the control logic are now dependent upon what instruction pattern is being applied to its inputs (i.e. on what instruction was fetched during the fetch cycle).

When the control logic has generated the required control signals to execute the instruction, it automatically performs another fetch cycle to retrieve the next instruction from memory and the cycle continues.

7.3 The Internal Structure of a Simple Machine

A diagram of the internal structure of a simple microprocessor is here:


A Simple Microprocessor
Control Bus
Address Bus
Data Bus
MAR
MDR
CR
Control Lines
IR
Control Logic
Internal Bus
B
C
D
E
PC
DP
SP
PSW
A
Tmp R
ALU

6. **The Control Register: (CR)** this is loaded only by the control logic, and is used to provide the necessary control signals to allow the memory and other I/O devices to be read to or written from. It allows the control logic to synchronise the data flow between the memory and the CPU.

7. **The Data Pointer (DP):** a register used to contain the address of any operand data that is required by the instruction. It is normally set up in the course of the opcode or operand fetch cycles, and is used during the execution cycle; its contents are loaded into the MAR during the instruction execution.

8. **The Stack Pointer (SP):** this is a dedicated register used to point to the top of the system stack.

9. **Processor Status Word (PSW):** is a word that stores the flags that indicate the status and error codes associated with the processor.

10. **The General Registers (B, C, D, E):** these registers are the general purpose working registers of the CPU. They are assumed to be the same width as the accumulator.

NOTE: In this simple machine, we assume that the PC has circuitry to automatically increment the register contents, and the stack pointer has circuitry to both increment and decrement its contents. The relevance of this will be seen later.

7.4 Microprocessor Instructions

Each instruction to the CPU will involve the processing or manipulation of data within the CPU in some way - transferring, adding, comparing etc. As such, each instruction must contain the following five pieces of information:

1. **The operation to be performed** (e.g. addition, data transfer, load, store etc.). This part of the instruction is referred to as the 'opcode'. A typical microprocessor will have a fairly limited set of primitive operations available - 16 or fewer.

2. **The source of the data**, or where the data is to be found. If the data is in memory, then the address must be specified; if the data is elsewhere, its location must be given in some other way.

3. **The source of the second data item**, for those instructions that require it. (e.g. addition requires two data items; complementing requires only one).

4. **The destination of the resultant data** after the operation has been performed.

5. Where the **next instruction**, to be executed after this one is complete, is to be found.

Clearly this cannot be contained in a single (typically 8-bit) memory location, and consequently it is necessary to reduce the instruction length in some way. Each of the five pieces of information cannot be omitted, so the reduction is achieved by making assumptions for some of them.

1. Requirement (5) may be omitted by assuming that the next instruction which is to be executed follows immediately after the current instruction in memory - in the next sequential memory location.

2. Requirements (2) to (4) can be eliminated by making use of internal registers in the CPU to contain the data (or the address of the data) and making their use implicit in a particular instruction. This increases the number of possible instructions, but by judicious choice it is possible to keep the total number of instructions below 256 and thus representable by a single 8-bit byte.

   As an example, the 8085 instruction:

   ADD C (instruction = 81H, or 10000001 binary)

   Specifies that:

1. the operation is addition
2. the source of the first data byte is the accumulator (implied)
3. the source of the second data byte is register C (implied)
4. the destination is the accumulator (implied)

The complete instruction thus takes only 8 bits to specify and may thus be contained in a single memory location.

Even after these reductions have been achieved, it is still necessary for some instructions to have information specified in addition to the opcode. Such an instruction will typically involve a memory access, and thus is required to give the address of the memory location being accessed. In this case, it is necessary to use more than 8 bits to specify the instruction; such an instruction is called a **multibyte instruction**.

The first byte of a multibyte instruction is the opcode. The additional bytes which give the extra information form what is called the operand. The whole sequence of bytes is the instruction.

7.5 Microcycles

The execution of each instruction in the CPU is made up of a number of smaller operations, called micro-operations or **microcycles**. Each microcycle takes one clock cycle to perform, and represents one data transfer operation or one register manipulation operation. It is represented by the following notation:

Data transfer operation:	**destination <- source**
For example:	MAR <- PC
	IR <- MDR
Register Manipulation Operation:	**destination <- expression**
For example:	PC <- PC + 1
	SP <- SP - 1

If it is necessary to transfer data to or from memory, then this cannot be done by a single microcycle. The external memory device is regarded as a series of registers, like the internal registers of the CPU; however, before one of these registers (words) can be read or written, it is necessary to indicate which register is to be affected. This is done by first placing the word's address in the Memory Address Register. This then causes the appropriate memory word to be enabled and 'connected' to the Memory Data Register. The data may then be transferred between the MDR and the addressed word (the direction depending on whether a read or write is taking place). This data transfer must occur at least one clock cycle after the address of the word was written to the MAR, to give the memory time to stabilise. These microcycles are represented as follows:

MAR <- source register
MDR <- memory(MAR)

Each microcycle which involves a transfer of data between two of the internal registers of the CPU uses the internal bus; consequently only one such microcycle can occur during a single clock cycle. If a microcycle involves register manipulation, however, or uses the external memory or address buses, then it can occur at the same time as a register transfer.

As an example, take the instruction "Add the contents of the accumulator to the contents of memory location 2100, and leave the result of the addition in the accumulator".

Assuming that the opcode of this instruction is A2 (hex), one operand is required (the address of the memory location - in this case 2100). Assume that the operand is stored in the next two bytes after the opcode (the address is 16 bits long and the memory locations are only 8 bits, hence two bytes of memory must be used). Also assume that the least significant byte of the address (00) is stored at the lower address, i.e. the byte after the opcode.

The memory locations and the connection to the CPU will appear as follows:

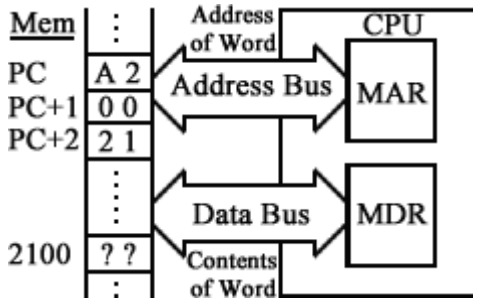


Figure 7.4: Memory Location and CPU Connection

The microcycles which would be involved on the simple machine illustrated in section 7.3 would be:

Cycle	Micro-operations	Description
I	MAR <- PC	Opcode address
IIa	PC <- PC + 1	PC points to the 1st operand
IIb	MDR <- memory (MAR)	Place opcode into MDR

III	IR <- MDR	Transfer opcode to IR
IV	MAR <- PC	Operand address into MAR
Va	PC <- PC + 1	PC points to 2nd operand
Vb	MDR <- memory (MAR)	Get 1st operand byte
VI	DP(L) <- MDR	Put into LSByte of DP
VII	MAR <- PC	2nd operand byte address to MAR
VIIIa	PC <- PC + 1	Point to next opcode
VIIIb	MDR <- memory (MAR)	Get 2nd operand byte
IX	DP(H) <- MDR	Put into MSbyte of DP
X	MAR <- DP	Address of operand data into MAR
XI	MDR <- memory (MAR)	Get operand data byte
XII	tempR <- MDR	Put into tempR of ALU
XIII	A <- A plus tempR	Use ALU to add A and tempR

1. Note that four machine cycles were used to perform this instruction. The first three are fetch cycles, since bytes that are part of the instruction (opcode, operands) are fetched from memory so that the instruction can be subsequently executed. The PC is used to point to the next byte to be fetched, and is incremented by 1 automatically after every memory fetch.

2. The PC is incremented during the same microcycle as the data is transferred to the MDR from the memory. This is because the PC has logic built into it to increment the contents, and thus does not require the internal bus or ALU. Note that the incrementing of the PC must occur in the cycle following its transfer to the MAR - it cannot be incremented and transferred at the same time.

3. A total of 13 clock cycles are used. Each clock cycle indicates one micro-operation, or internal register transfer.

4. The first fetch (the opcode fetch cycle) is the same for all instructions. Thereafter any further fetch cycles are dictated by the opcode itself, which is now in the IR and influences the operation of the control logic.

Summarizing: The whole cycle is an '**instruction cycle**'. This is composed of two sections:

1. Fetch the instruction, and any necessary operands, using one or more fetch cycles. The PC is incremented automatically after every fetch cycle.

2. Execute the instruction.