CPU Structure and Functions*

Nguyen Thi Hoang Lan

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

This module present the processor organization. We will begin with a summary of processor organization. Registers, which form the internal memory of the processor, are then analyzed. We are then in a position to return to the discussion of the instruction cycle. A description of the instruction cycle and a common technique known as instruction pipelining completes our description. Understanding the organization of the CPU of computer

Processor Organization

To understand the organization of the CPU, let us consider the requirements placed on the CPU, the things that it must do:

- **Fetch instruction**: The CPU reads an instruction from memory.
- **Interpret instruction**: The instruction is decoded to determine what action is required.
- **Fetch data**: The execution of an instruction may require reading data from memory or an I/O module.
- **Process data**: The execution of an instruction may require performing some arithmetic or logical operation on data.
- **Write data**: The results of an execution may require writing data to memory or an I/O module.

To do these things, it should be clear that the CPU needs to store some data temporarily. It must remember the location of the last instruction so that it can know where to get the next instruction. It needs to store instructions and data temporarily while an instruction is being executed. In other words, the CPU needs a small internal memory.

Figure 1 is a simplified view of a CPU, indicating its connection to the rest of the system via the system bus. You will recall (Lecture 1) that the major components of the CPU are an **arithmetic and logic unit** (ALU) and a **control unit** (CU). The ALU does the actual computation or processing of data. The control unit controls the movement of data and instructions into and out of the CPU and controls the operation of the ALU. In addition, the figure shows a minimal internal memory, consisting of a set of storage locations, called **registers**.

*Version 1.1: Jul 15, 2009 9:09 pm -0500

†http://creativecommons.org/licenses/by/3.0/

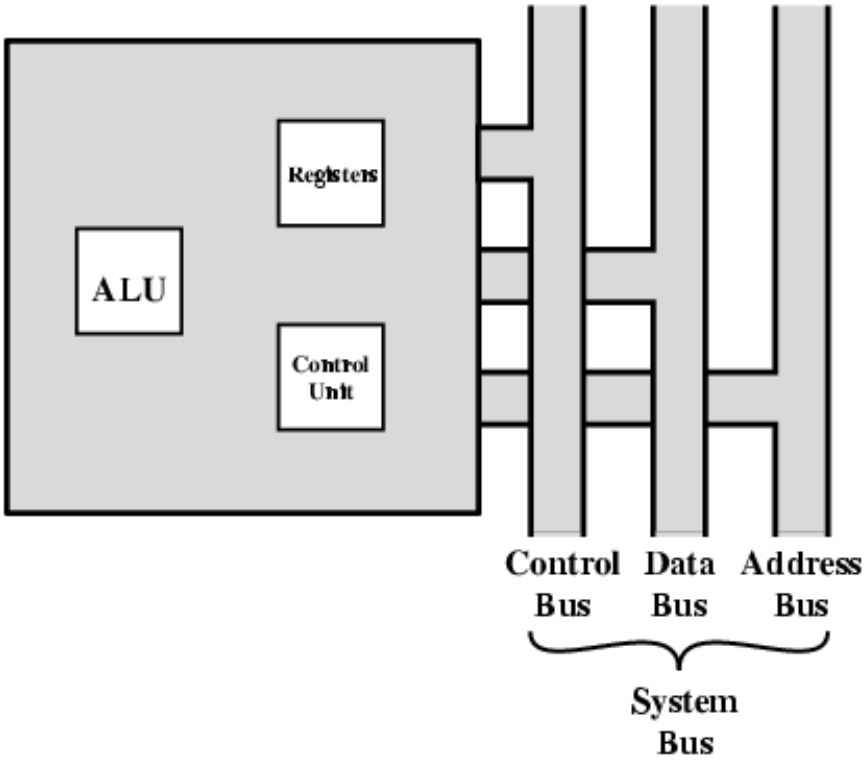

Figure 1: The CPU with the System Bus

Figure 2 is a slightly more detailed view of the CPU. The data transfer and logic control paths are indicated, including an element labeled **internal CPU-bus**. This element is needed to transfer data between the various registers and the ALU because the ALU in fact operates only on data in the internal CPU memory.

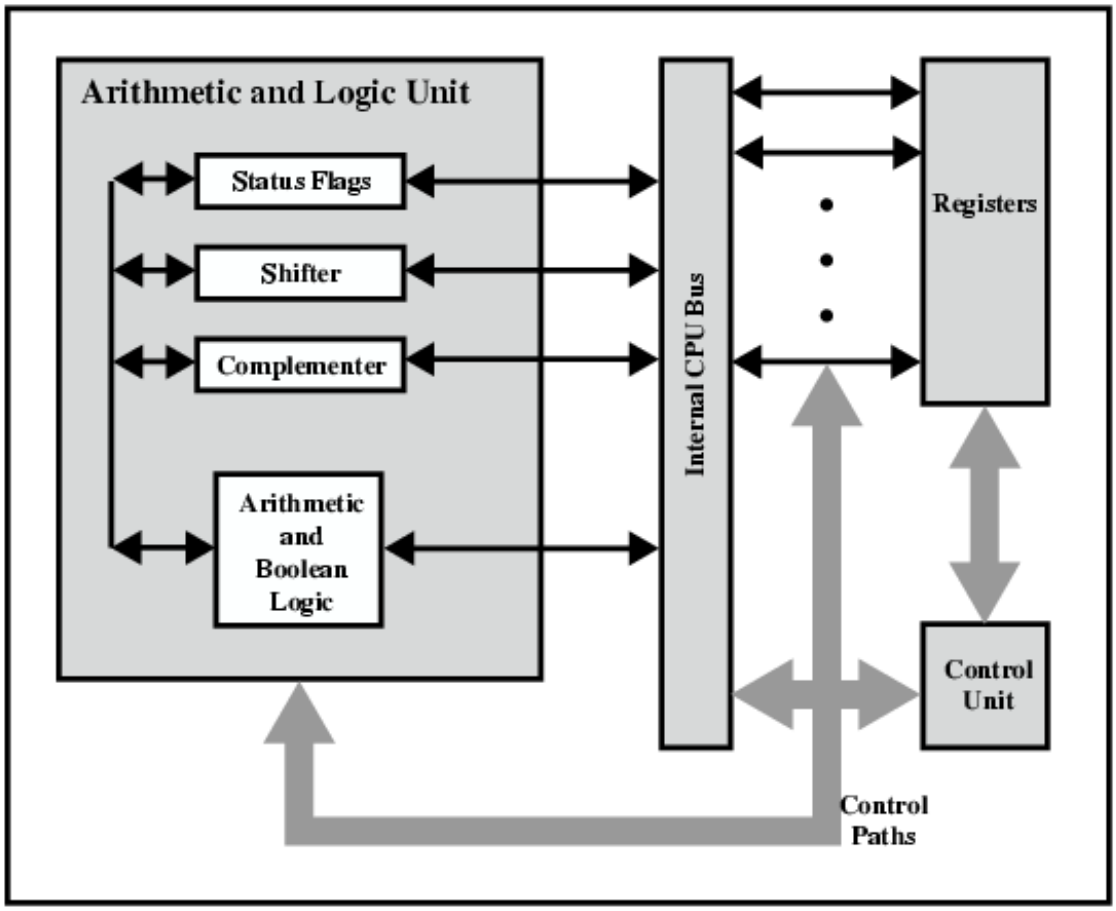

Figure 2: CPU Internal Structure

1 Register organization

Within the CPU, there is a set of registers that function as a level of memory above main memory and cache in the hierarchy. The registers in the CPU perform two roles:

- **User-visible registers**: These enable the machine- or assembly-language programmer to minimize main memory references by optimizing use of registers.
- **Control and status registers**: These are used by the control unit to control the operation of the CPU and by privileged, operating system programs to control the execution of programs.

There is not a clean separation of registers into these two categories. For example, on some machines the program counter is user visible (e.g., Pentium), but on many it is not (e.g., PowerPC). For purposes of the following discussion, however, we will use these categories.

1.1 User-Visible Registers

A user-visible register is one that may be referenced by means of the machine language that the CPU executes. We can characterize these in the following categories:

- General purpose
- Data
- Address
- Condition codes

General-purpose registers: can be assigned to a variety of functions by the programmer. Sometimes their use within the instruction set is orthogonal to the operation. That is, any general–purpose register can contain the operand for any opcode. This provides true general-purpose register use. Often, however, there are restrictions. For example, there may be dedicated registers for floating-point and stack operations. In some cases, general-purpose registers can be used for addressing functions (e.g.. register indirect, displacement). In other cases, there is a partial or clean separation between data registers and address registers.

Data registers may be used only to hold data and cannot be employed in the calculation of an operand address.

Address registers may themselves be somewhat general purpose, or they may be devoted to a particular addressing mode. Examples include the following:

- **Segment pointers**: In a machine with segmented addressing, a segment register holds the address of the base of the segment. There may be multiple registers: for example, one for the operating system and one for the current process.
- **Index registers**: These are used for indexed addressing and may be autoindexed.
- **Stack pointer**: If there is user-visible stack addressing, then typically the stack is in memory and there is a dedicated register that points to the top of the slack. This allows implicit addressing; that is, push, pop, and other slack instructions need not contain an explicit stack operand.

Condition codes register (also referred to **as flags**): Condition codes are bits set by the CPU hardware as the result of operations. For example, an arithmetic operation may produce a positive, negative, zero, or overflow result. In addition to the result itself being stored in a register or memory, a condition code is also set. The code may subsequently be tested as part of a conditional branch operation.

1.2 Control and Status Registers

There are a variety of CPU registers that are employed to control the operation of the CPU. Most of these, on most machines, are not visible to the user. Some of them may be visible to machine instructions executed in a control or operating system mode.

Of course, different machines will have different register organizations and use different terminology. We list here a reasonably complete list of register types, with a brief description.

Four registers are essential to instruction execution:

- **Program counter (PC)**: Contains the address of an instruction to be fetched.
- **Instruction register (IR)**: Contains the instruction most recently fetched.
- **Memory address registers (MAR)**: Contains the address of a location in memory.
- **Memory buffer register (MBR)**: Contains a word of data lo be written to memory or the word most recently read.

Typically, the CPU updates the PC after each instruction fetch so that the PC always points to the next instruction to be executed. A branch or skip instruction will also modify the contents of the PC. The fetched instruction is loaded into an IR, where the opcode and operand specifiers are analyzed. Data are exchanged with memory using the MAR and MBR. In a bus-organized system, the MAR connects directly to the

address bus, and the MBR connects directly to the data bus. User-visible registers, in turn, exchange data with the MBR.

The four registers just mentioned are used for the movement of data between the CPU and memory. Within the CPU, data must be presented to the ALU for processing. The ALU may have direct access to the MBR and user-visible registers. Alternatively, there may be additional buffering registers at the boundary to the ALU: these registers serve as input and output registers for the ALL and exchange data with the MBR and user-visible registers.

All CPU designs include a register or set of registers, often known as the **program status word** (PSW), that contain status information. The PSW typically contains condition codes plus other stains information. Common fields or flags include the following:

- **Sign**: Contains the sign bit of the result of the last arithmetic operation.
- **Zero**: Set when the result is 0.
- **Carry**: Set if an operation resulted in a carry (addition) into or borrow (sub-traction) out of a high-order hit. Used for multiword arithmetic operations.
- **Equal**: Set if a logical compare result is equality.
- **Overflow**: Used to indicate arithmetic overflow
- **Interrupt enable/disable**: Used to enable or disable interrupts.
- **Supervisor**: Indicates whether the CPU is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode, and certain areas of memory can be accessed only in supervisor mode.

A number of other registers related to status and control might be found in a particular CPU design. In addition to the PSW, there may be a pointer to a block of memory containing additional status information (e.g., process control blocks).

Example Register Organizations:

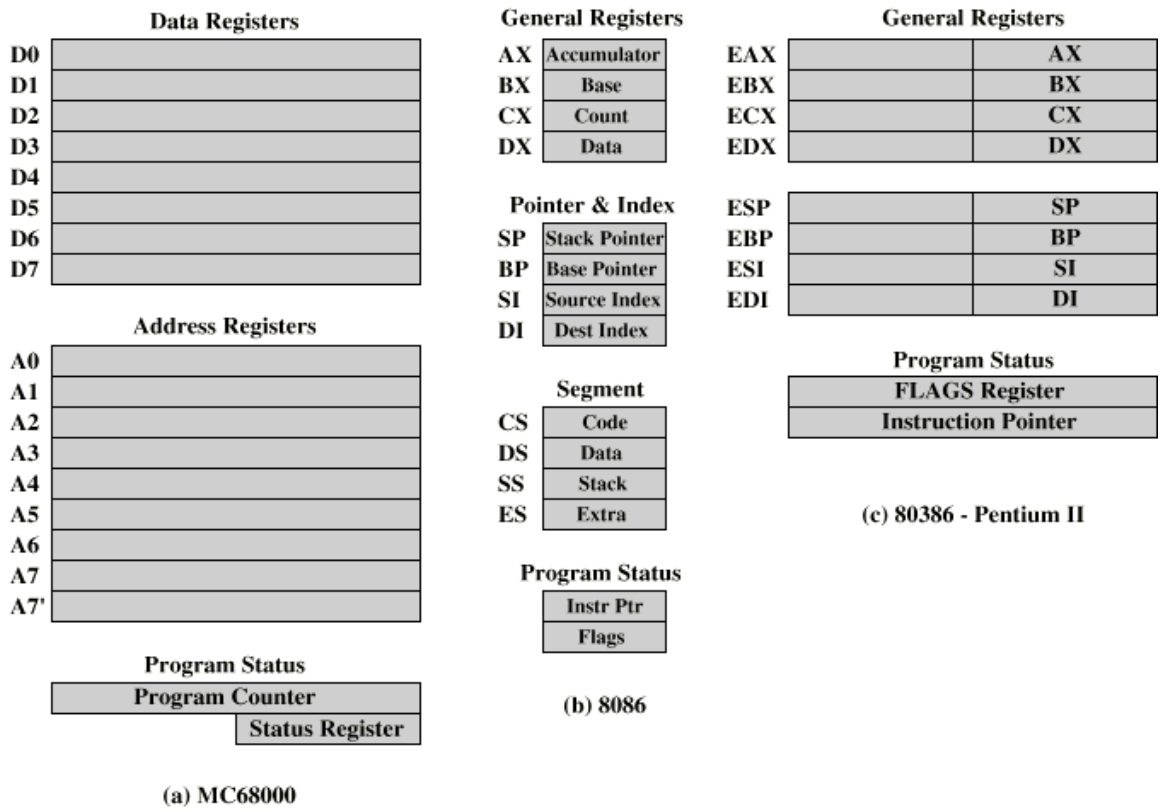

Figure 3: Example of microprocessor registers organizations