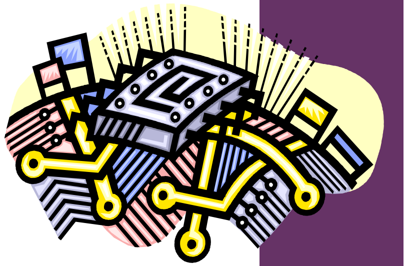
+

Lecture B - 02

Structure and Function of Central Processing Unit, Instruction Cycle

Processor Organization

Processor Requirements:



- Fetch instruction
  - The processor reads an instruction from memory (register, cache, main 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
- In order to do these things the processor needs to store some data temporarily and therefore needs a small internal memory

CPU With the System Bus

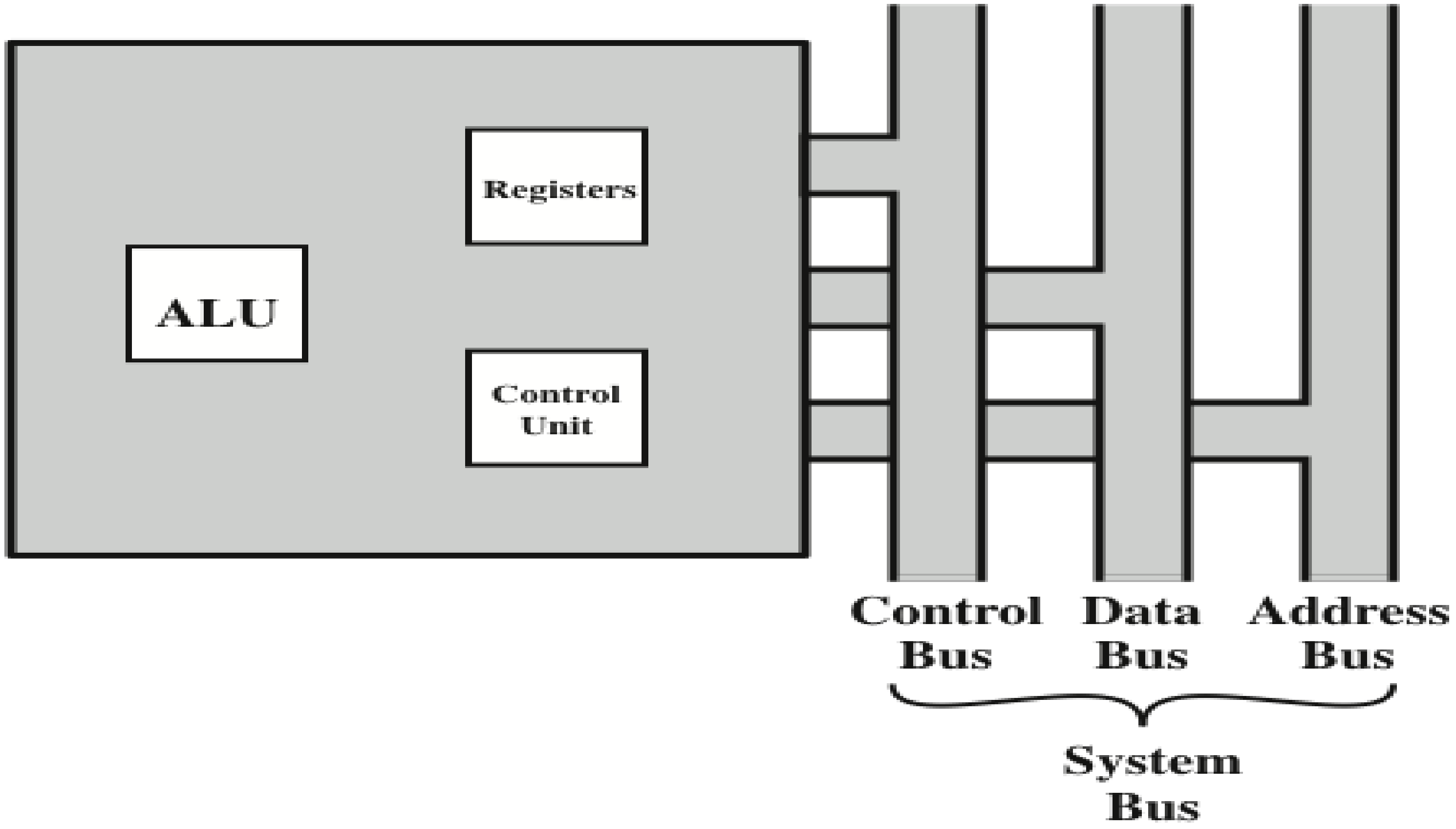


Figure 14.1 The CPU with the System Bus

CPU Internal Structure

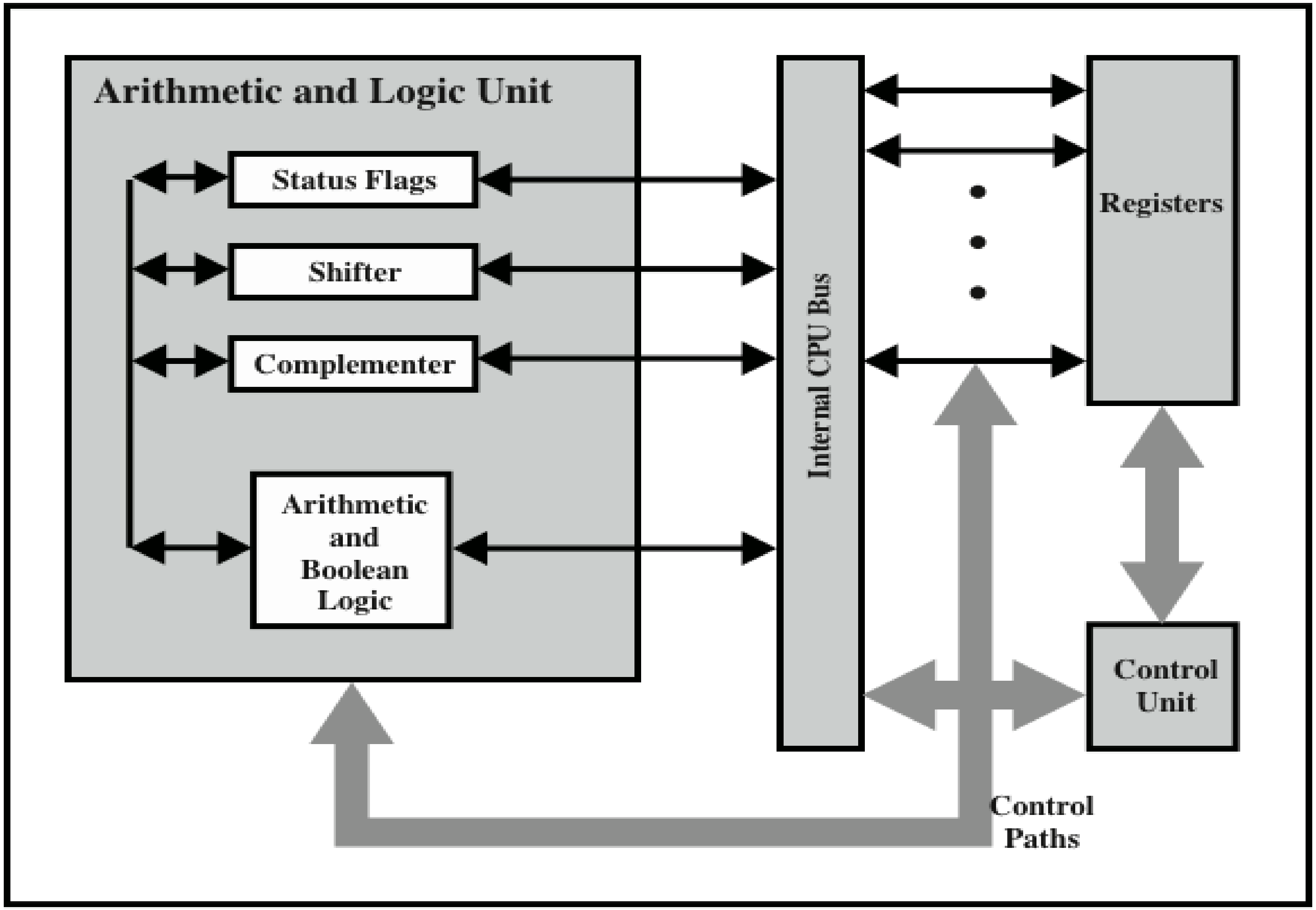


Figure 14.2 Internal Structure of the CPU

+

Register Organization

- Within the processor 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 processor perform two roles:

User-Visible Registers	Control and Status Registers
Enable the machine or assembly language programmer to minimize main memory references by optimizing use of registers	Used by the control unit to control the operation of the processor and by privileged operating system programs to control the execution of programs

User-Visible Registers

Referenced by means of the machine language that the processor executes

Categories:

- **General purpose**
  - Can be assigned to a variety of functions by the programmer
- **Data**
  - May be used only to hold data and cannot be employed in the calculation of an operand address
- **Address**
  - May be somewhat general purpose or may be devoted to a particular addressing mode
  - Examples: segment pointers, index registers, stack pointer
- **Condition codes**
  - Also referred to as *flags*
  - Bits set by the processor hardware as the result of operations

+
Control and Status Registers

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 register (MAR)
  - Contains the address of a location in memory
- Memory buffer register (MBR)
  - Contains a word of data to be written to memory or the word most recently read

+

Program Status Word (PSW)

Register or set of registers that contain status information

Common fields or flags include:

- Sign
- Zero
- Carry
- Equal
- Overflow
- Interrupt Enable/Disable
- Supervisor

Example Microprocessor Register Organizations

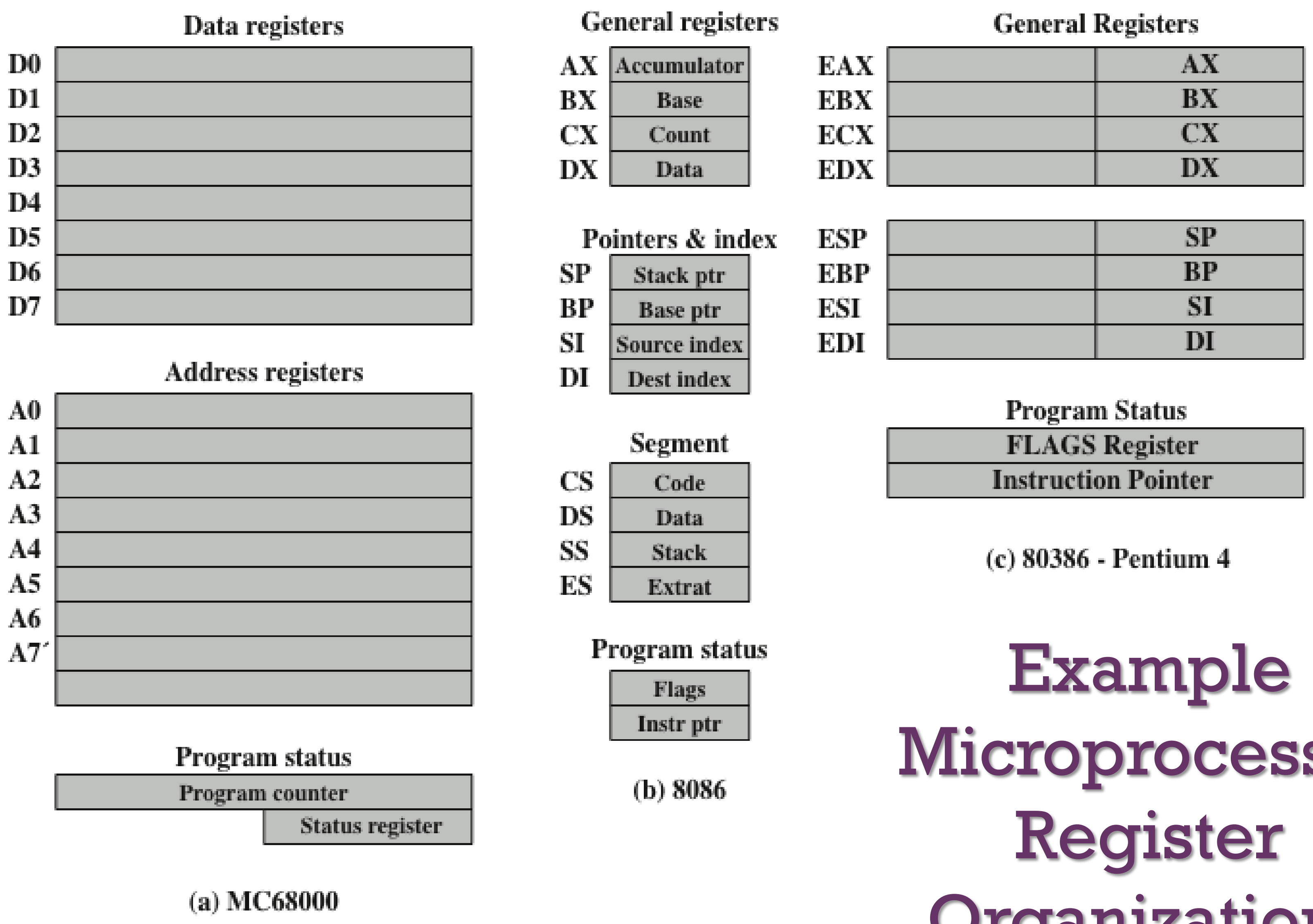


Figure 14.3 Example Microprocessor Register Organizations

Instruction Cycle

Includes the following stages:

- Fetch
  - Read the next instruction from memory into the processor
- Execute
  - Interpret the opcode and perform the indicated operation
- Interrupt
  - If interrupts are enabled and an interrupt has occurred, save the current process state and service the interrupt

Instruction Cycle

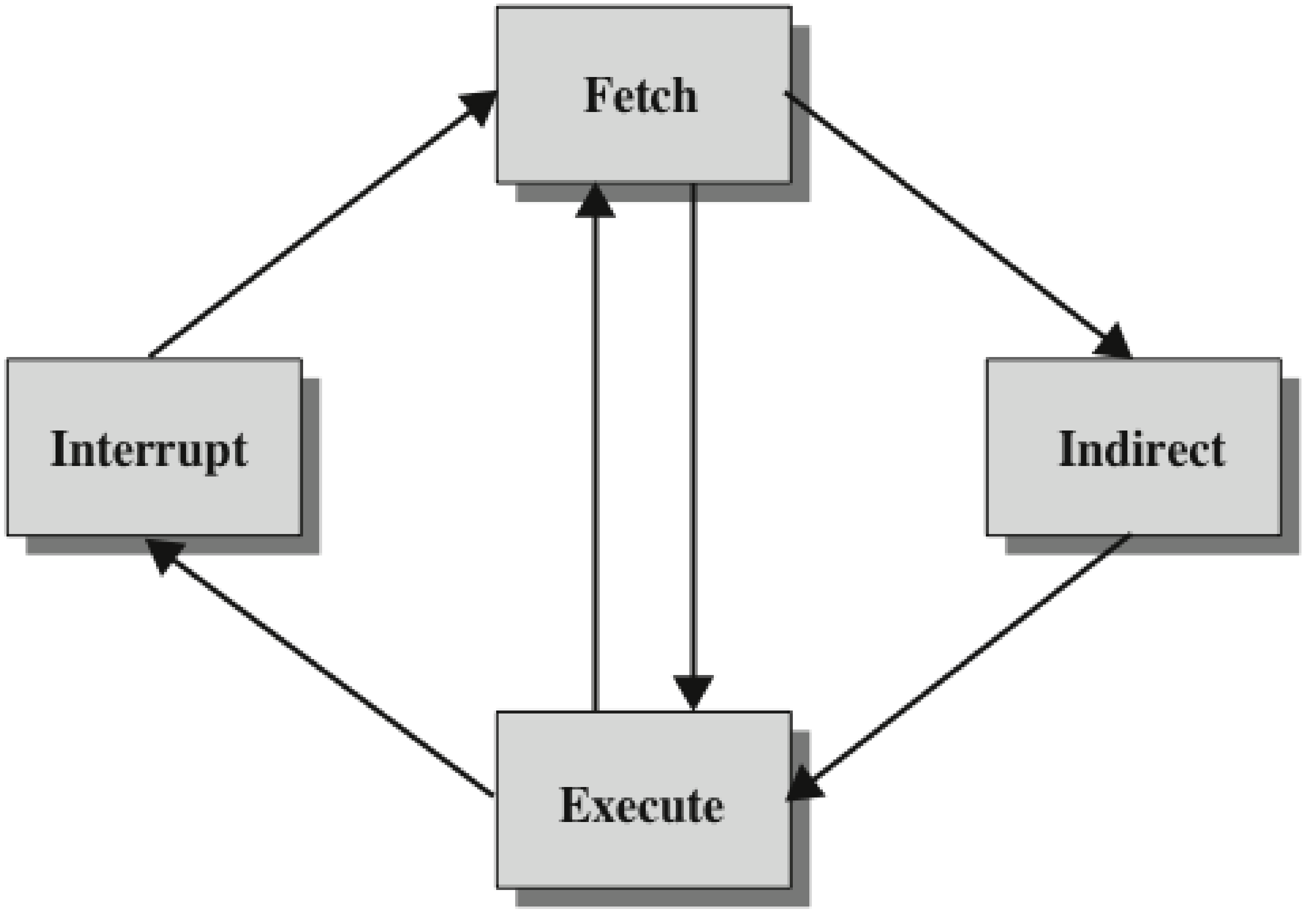


Figure 14.4 The Instruction Cycle

Instruction Cycle State Diagram

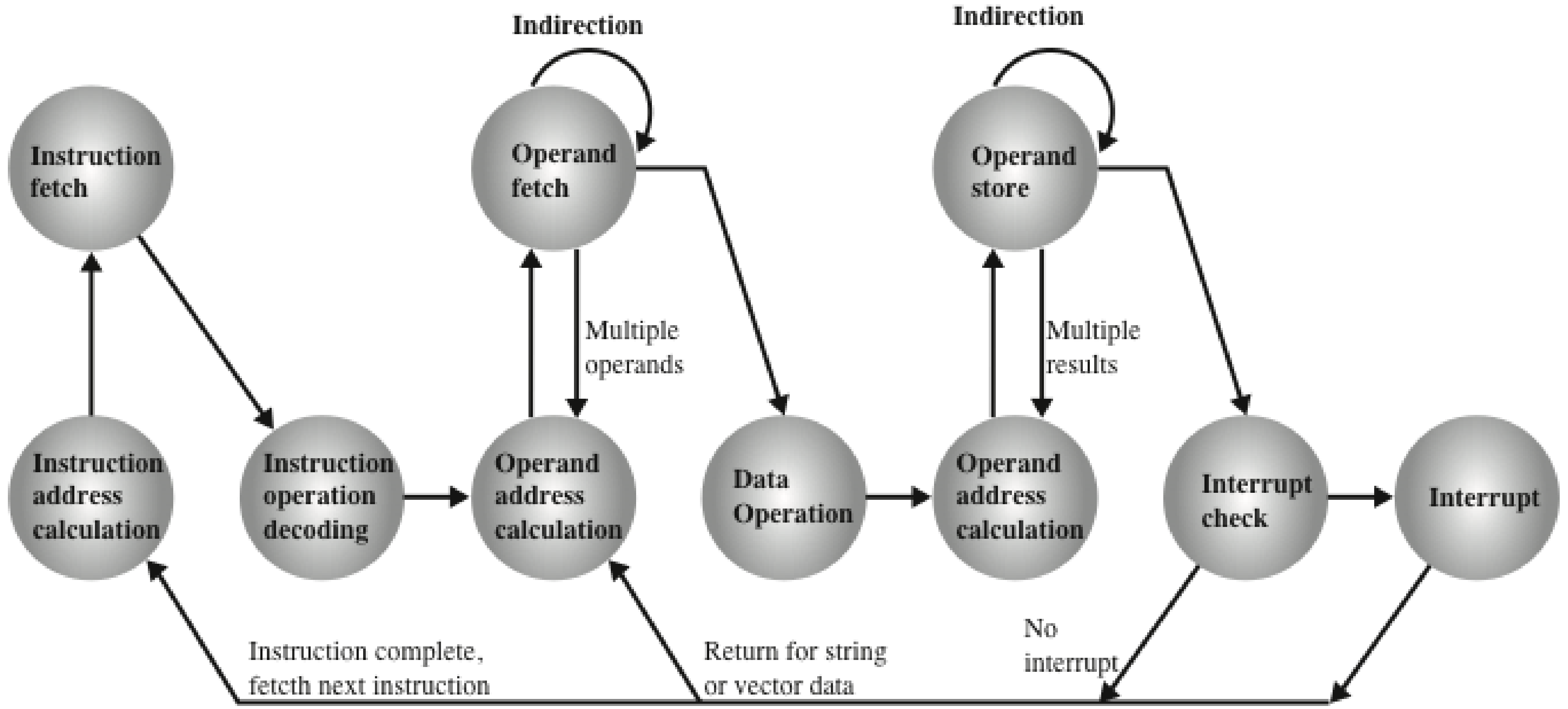
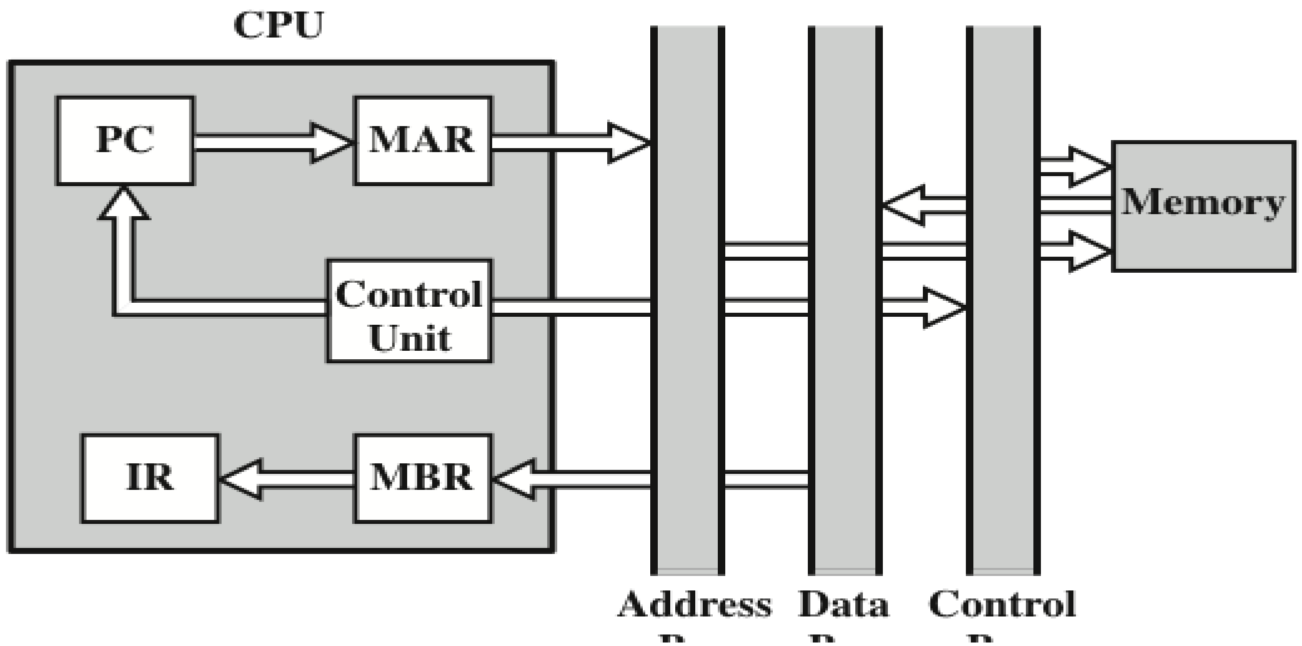

Indirection
Indirection
Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
No interrupt
Instruction complete, fetch next instruction
Return for string or vector data

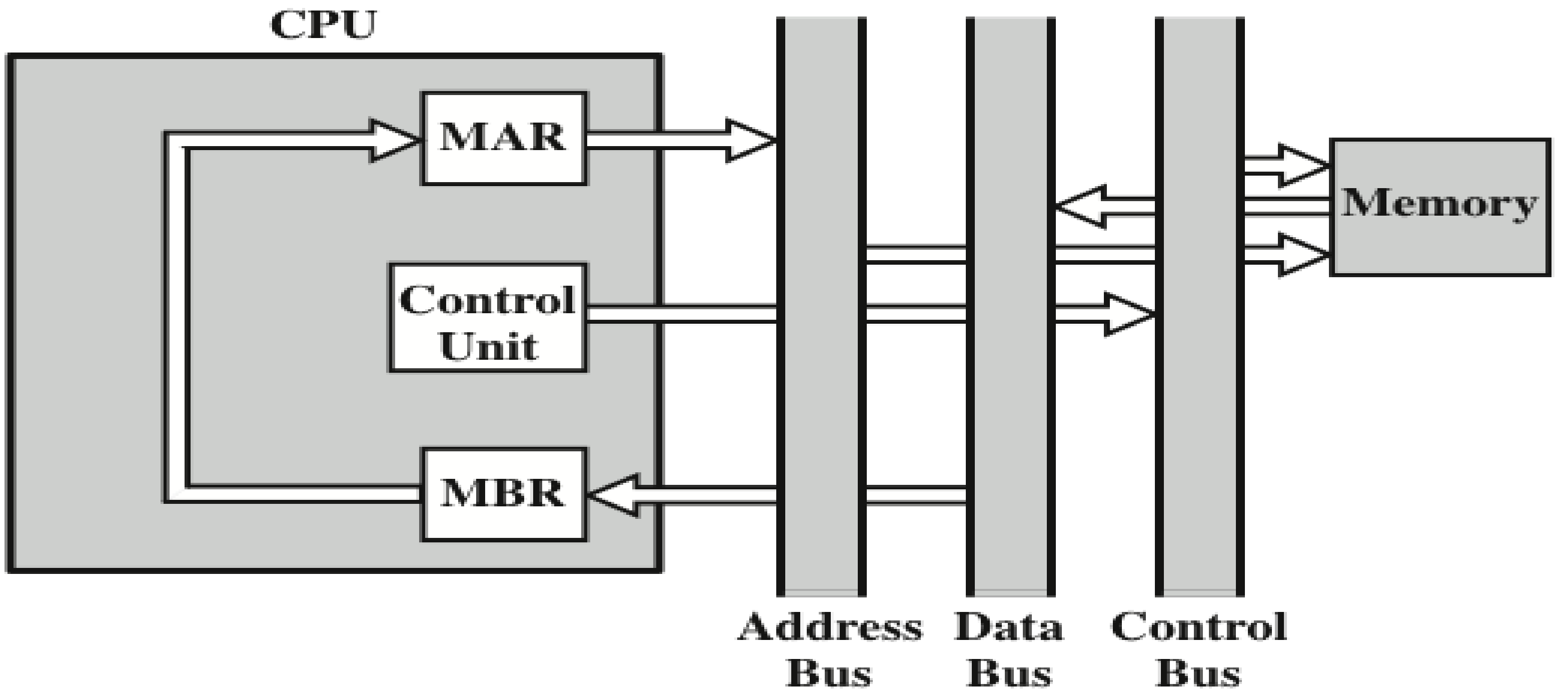
Figure 14.5 Instruction Cycle State Diagram

Data Flow, Fetch Cycle



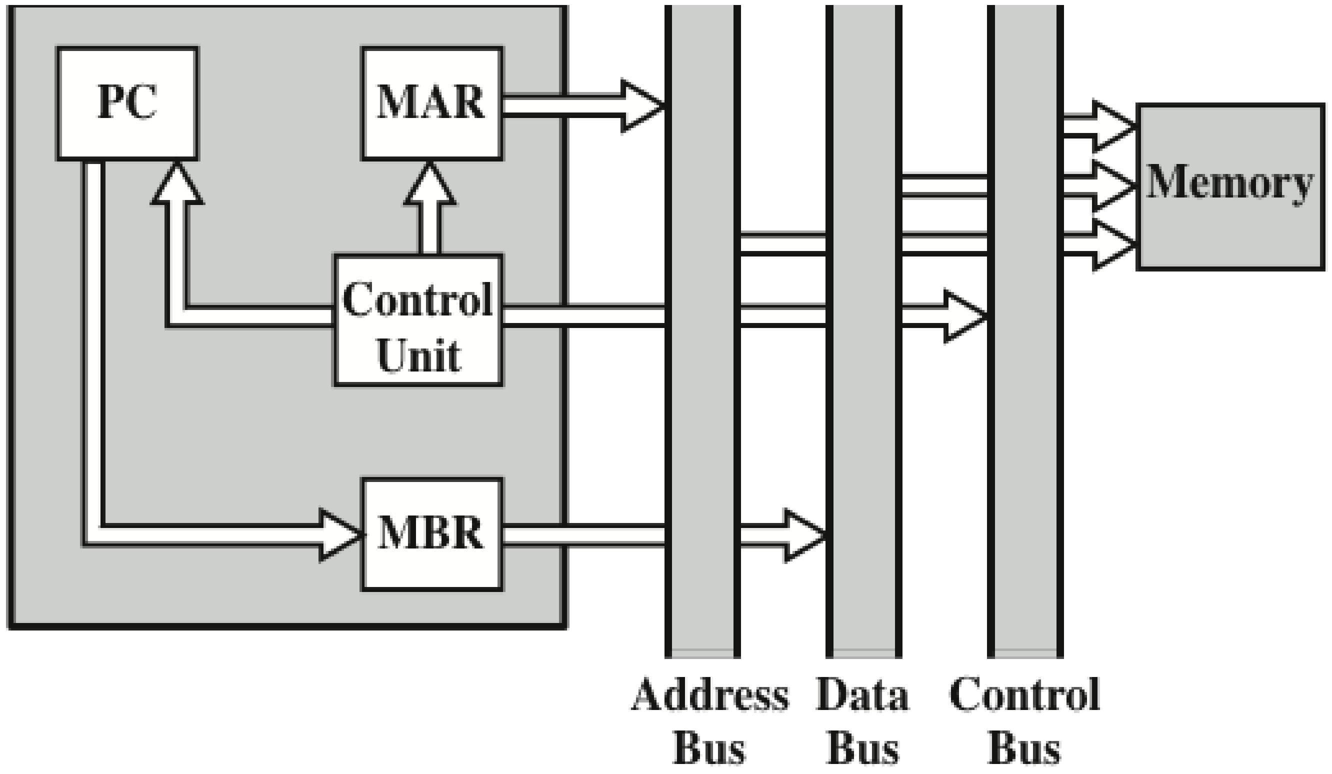
The PC contains the address of the next instruction to be fetched. This address is moved to the MAR and placed on the address bus. The control unit requests a memory read, and the result is placed on the data bus and copied into the MBR and then moved to the IR. Meanwhile, the PC is incremented by 1, preparatory for the next fetch.

Data Flow, Indirect Cycle



The right- most N bits of the MBR, which contain the address reference, are transferred to the MAR. Then the control unit requests a memory read, to get the desired address of the operand into the MBR.

Data Flow, Interrupt Cycle



The current contents of the PC must be saved so that the processor can resume normal activity after the interrupt. Thus, the contents of the PC are transferred to the MBR to be written into memory. The special memory location reserved for this purpose is loaded into the MAR from the control unit. It might, for example, be a stack pointer. The PC is loaded with the address of the interrupt routine. As a result, the next instruction cycle will begin by fetching the appropriate instruction.

+

Example – Intel 8086 (Register Organizations) (1)

- The model includes fourteen 16-bit registers:
  - Four segment registers (CS, DS, SS and ES)
  - One Instruction pointer
  - Four data registers (AX, BX, CX and DX)
  - Two pointer registers (BP and SP)
  - Two index registers (SI and DI)
  - One Status Register (SR), with nine of its bits implemented as status and control flags
- The 8086 architecture implements independent memory and input/output address spaces
- The memory address space is 2^{20} = 1,048,576 bytes (1Mbytes) in size and the I/O address space is 2^{16} = 65,536 bytes (64Kbytes) in size

+

Example - The ARM Processor

ARM is primarily a RISC system with the following attributes:

- Moderate array of uniform registers
- A load/store model of data processing in which operations only perform on operands in registers and not directly in memory
- A uniform fixed-length instruction of 32 bits for the standard set and 16 bits for the Thumb instruction set
- Separate arithmetic logic unit (ALU) and shifter units
- A small number of addressing modes with all load/store addresses determined from registers and instruction fields
- Auto-increment and auto-decrement addressing modes are used to improve the operation of program loops
- Conditional execution of instructions minimizes the need for conditional branch instructions, thereby improving pipeline efficiency, because pipeline flushing is reduced

Characteristics of Reduced Instruction Set Architectures

One machine instruction per machine cycle	• *Machine cycle* --- the time it takes to fetch two operands from registers, perform an ALU operation, and store the result in a register
Register-to-register operations	• Only simple LOAD and STORE operations accessing memory • This simplifies the instruction set and therefore the control unit
Simple addressing modes	• Simplifies the instruction set and the control unit
Simple instruction formats	• Generally only one or a few formats are used • Instruction length is fixed and aligned on word boundaries • Opcode decoding and register operand accessing can occur simultaneously

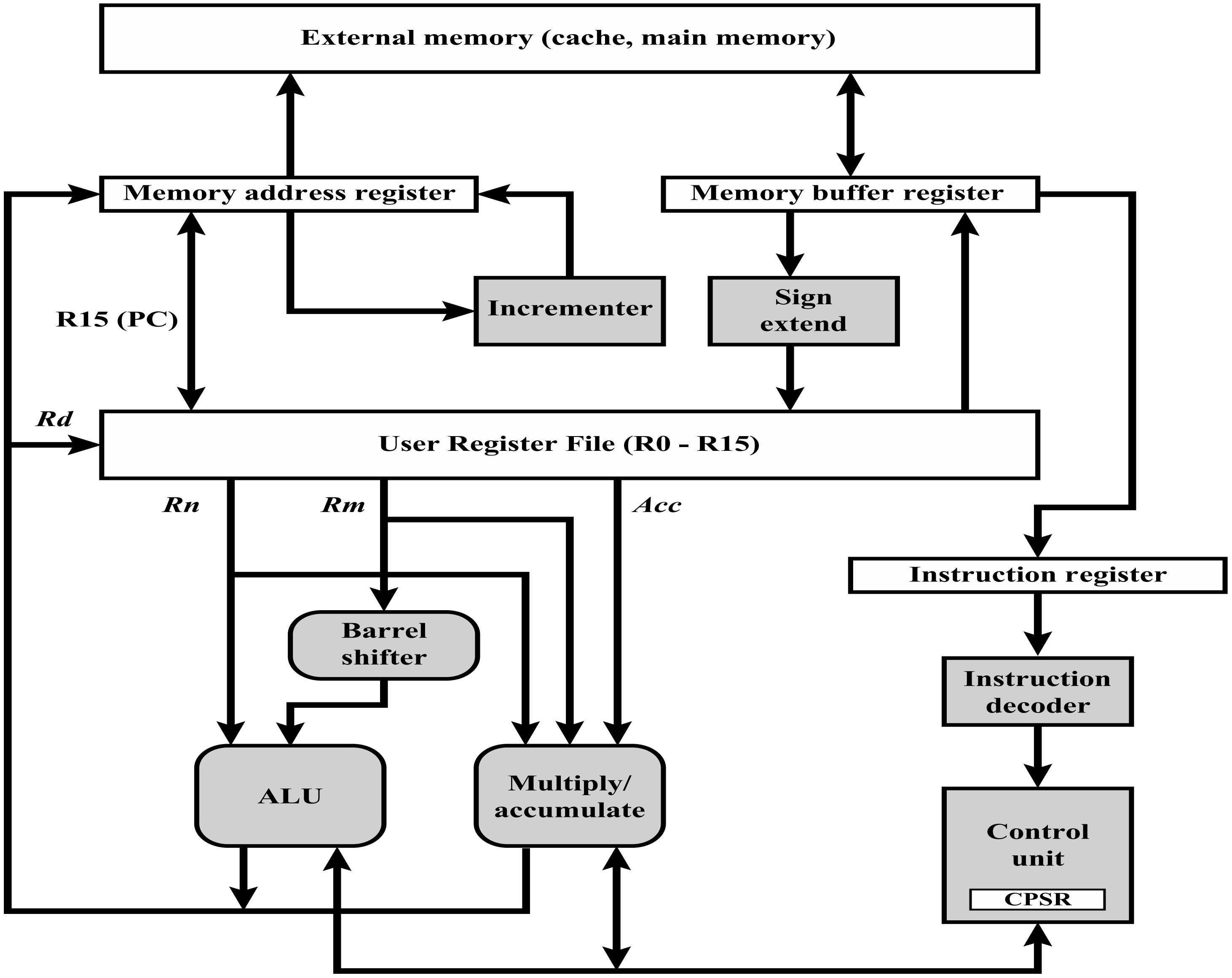


Figure 14.25 Simplified ARM Organization

Processor Modes

ARM architecture supports seven execution modes

Most application programs execute in user mode

- While the processor is in user mode the program being executed is unable to access protected system resources or to change mode, other than by causing an exception to occur

Remaining six execution modes are referred to as privileged modes

- These modes are used to run system software

Advantages to defining 6 different privileged modes

- **The OS can tailor the use of system software to a variety of circumstances**
- Certain registers are dedicated for use for each of the privileged modes, allows swifter changes in context

Privileged mode - 5 Exception Modes and 1 system mode

Have full access to system resources and can change modes freely

Entered when specific exceptions occur

Exception modes:

- Supervisor mode
- Abort mode
- Undefined mode
- Fast interrupt mode
- Interrupt mode

System mode:

- Not entered by any exception and uses the same registers available in User mode
- Is used for running certain privileged operating system tasks
- May be interrupted by any of the five exception categories

User-visible registers for the ARM

Modes						
	Privileged modes					
		Exception modes				
User	System	Supervisor	Abort	Undefined	Interrupt	Fast Interrupt
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13 (SP)	R13 (SP)	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14 (LR)	R14 (LR)	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Shading indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode.

SP = stack pointer	CPSR = current program status register
LR = link register	SPSR = saved program status register
PC = program counter	

+ Example – Intel 8086 (Register Organizations)

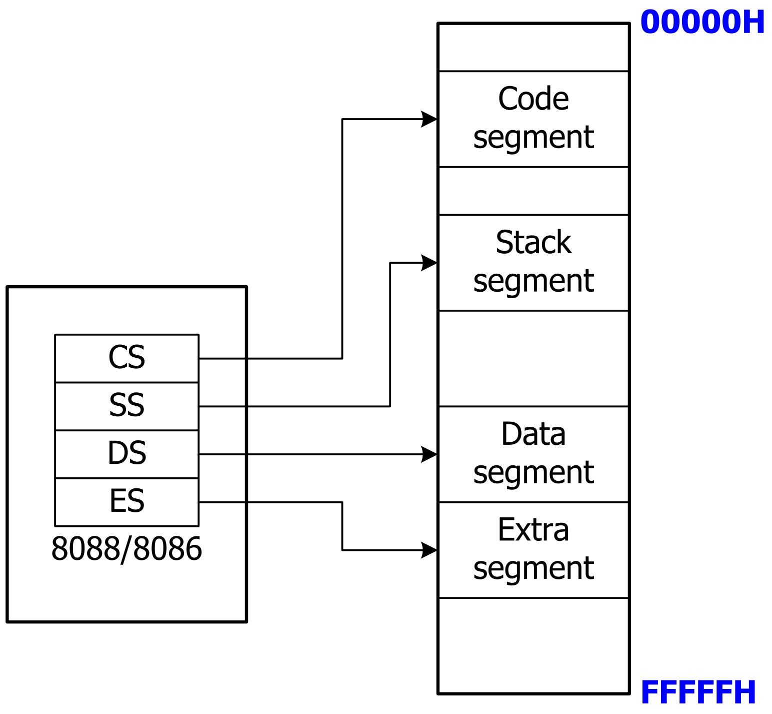
Segment Registers & Memory segmentation

- The 8086 has 1Mbyte address space, but not all is active at one time
- The 1Mbyte of memory is partitioned into segments
- One segment represents an independently addressable memory locations of 64K consecutive bytes (64Kbytes per segment)
- Each segment is assigned a base address (starting address)
- Only 4 segments are active at one time: code segment, stack segment, data segment, extra segment
- The segments that are active are identified by the values of four segment registers:

CS, SS, DS and ES

+ Example – Intel 8086 (Register Organizations)

- This gives a maximum of 256Kbytes of active memory:
  i. 64Kbytes for program storage (code segment)
  ii. 64Kbytes for stack (stack segment)
  iii. 128Kbytes for data storage (data and extra segment)



+ Example – Intel 8086 (Register Organizations) (4)

- **CS (Code Segment) register**

  - selects the segment from which instructions of the program are fetched and executed
- **SS (Stack Segment) register**

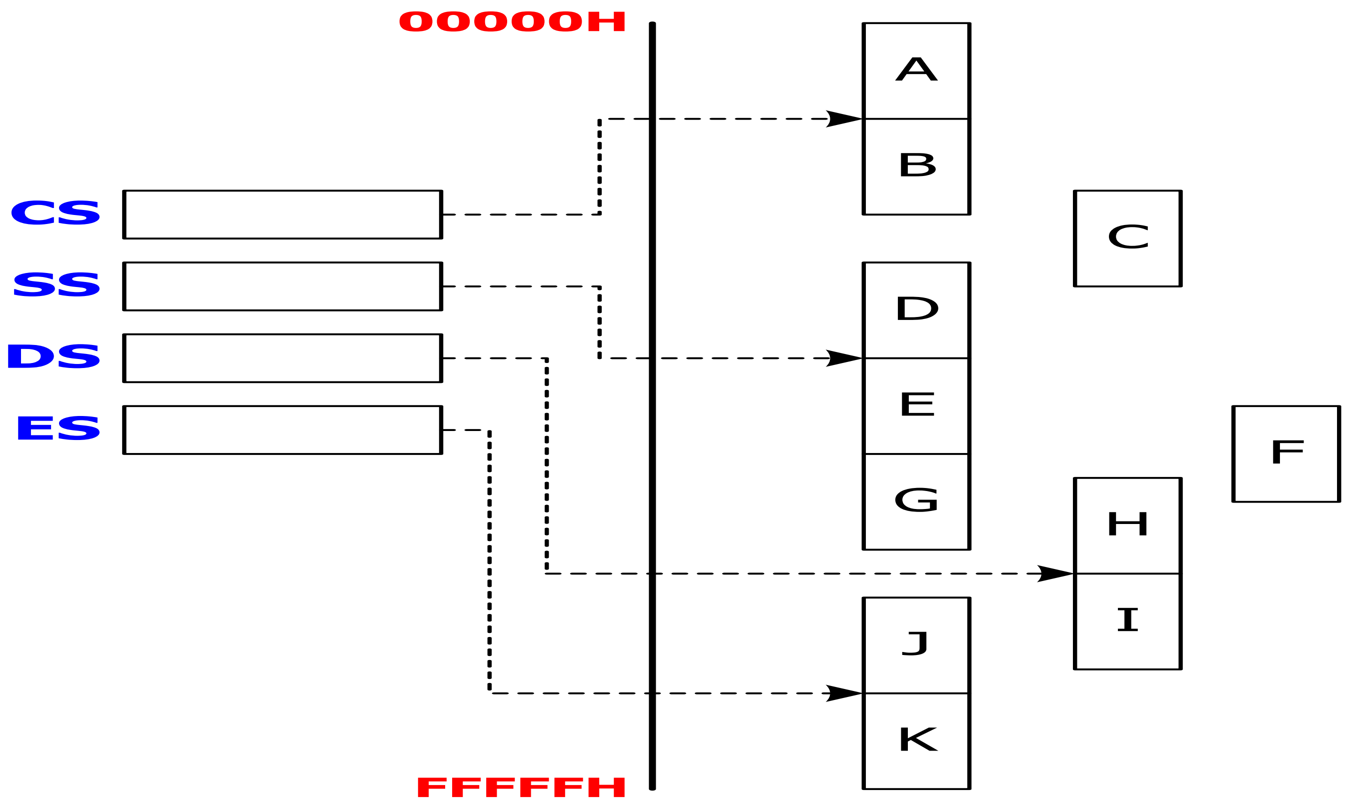
  -selects the segment to be used as a stack
- **DS (Data Segment) register**

  -selects the segment in which data are to be processed by the MPU are stored
- **ES (Extra Segment) register**

  -selects a second 64Kbyte segment for data storage
- The segment registers are user accessible (their value can be changed using software)
- Therefore, to gain access to another part of memory, simply change the value of the appropriate registers

+ Example – Intel 8086 (Register Organizations) (5)

- Segments can be contiguous (adjacent), disjointed or overlapping
- For e.g., A and B are contiguous, B & C are overlapping, B and D are disjointed



Example – Intel 8086 (Register Organizations)

- One restriction of the segment base address: only the upper 4 hexadecimal digit of the address can be changed (these 4 digits are the values inside the segment registers)

- Examples of valid base addresses are 00000H, 00010H, 00020H etc.

- E.g. Determine the segment base addresses of the code segment, data segment, extra segment and stack segment if the contents of the segment registers are as shown below.

CS	0120H
DS	0250H
ES	0500H
SS	2020H

+ Example – Intel 8086 (Register Organizations) (7)

Instruction Pointer

- Instruction Pointer (IP) is a 16-bit register that identifies the location of the next instruction to be fetched from the current code segment (depends on CS)
- It contains the offset of the next instruction instead of its actual address
- IP and CS are both 16-bit registers, but a 20-bit address is needed to access memory
- The offset in IP is combined with the code segment base address in CS to generate the address of the next instruction (denoted as CS:IP)
- Every time an instruction is fetched from memory, 8088/8086 updates its IP by incrementing it by two (to the address of the next instruction)
- To change the active code segment, just load the new value in CS register

+ Example – Intel 8086 (Register Organizations) (8)

Data Registers

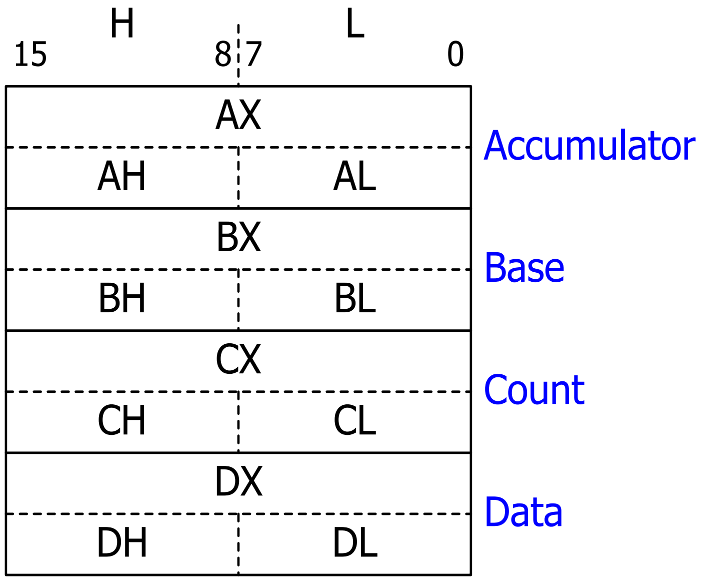
- 8088/8086 has four general purpose data registers:
  - A - Accumulator register
  - B - Base register
  - C - Count register
  - D - Data register
- During program execution, they hold the frequently accessed values or results
- All general purpose data registers can be used as the source or destination of an operand during arithmetic operations (e.g. ADD) or logic operations (e.g. AND).
- The advantage of storing these data in internal registers instead of memory is that they can be accessed much faster

+ Example – Intel 8086 (Register Organizations) (9)

- Each of these registers can either be accessed as a whole (16 bits) or as two 8-bit registers

- An X after the register letter identifies the reference of a register as a word (16 bits)

- On the other hand, when referencing one of these registers on 8-bit basis, the register name is followed by the letter H or L, identifies the high byte or low byte.



+ Example – Intel 8086 (Register Organizations) (10)

Pointer and Index Register

- There are four other general-purpose registers:
  i. Two pointer registers
     - Base Pointer (BP) and Stack Pointer (SP)
  ii. Two index registers
     - Source Index (SI) and Destination Index (DI)
- These registers store offset address, which represents the displacement of a storage location in memory from the segment base address (value in segment registers)
- They are used as pointer or index to select a specific location within a 64Kbyte segment
- Index registers are used to refer to data in memory relative to data segment or extra segment

+ Example – Intel 8086 (Register Organizations) (11)

- Pointer registers are used to access memory locations relative to the stack segment
- The value held inside these registers can be read, loaded or modified through software
- Unlike data registers, the pointer and index registers can only be accessed on 16-bit basis

+ Example – Intel 8086 (Register Organizations) (12)

Status Register

- The status register (also called flags register), is a 16-bit register within 8088/8086
- Only 9 bits of this register are used

X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

Flag register of 8086

+ Example – Intel 8086 (Register Organizations) (13)

- 6 status flags: carry flag (CF), parity flag (PF), auxiliary carry flag (AF), zero flag (ZF), sign flag (SF) and overflow flag (OF)

- The logic state of these flags indicate conditions that are produced as the result of executing an instruction
- **Carry flag (CF)**: CF is set (1) if there is a carry out or borrow in for the most significant bit of the result when executing an instruction; otherwise, CF is cleared (0)
- **Parity flag (PF)**: PF is set (1) if the result produced by an instruction has even parity (even number of 1); otherwise, PF is cleared (0)
- **Auxiliary carry flag (AF)**: AF is set (1) if there is a carry-out from the low nibble into the high nibble or a borrow-in from the high nibble into the low nibble of the lower byte in a 16-bit word; otherwise, AF is cleared (0)
- **Zero flag (ZF)**: ZF is set (1) if the result of an instruction is zero; otherwise, ZF is cleared (0)

+ # Example – Intel 8086 (Register Organizations) (14)

- **Sign flag (SF)**: The MSB of the result is copied into SF. Thus, SF is set (1) is the result is a negative number or cleared (0) if it is positive.
- **Overflow flag (OF)**: When OF is set, it indicates that the signed result is out of range. If the result is not out of range, OF remains reset
  - E.g. Consider executing the following instructions:

```
MOV AL, 75
ADD AL, 85
```

$$\begin{array}{ccccccccccc}
 & 1 & & 1 & 1 & 1 & 1 & 1 & & & \\
 & \underline{0} & 1 & 0 & 0 & 1 & 0 & 1 & 1 & & 75_{10} \\
+ & \underline{0} & 1 & 0 & 1 & 0 & 1 & 0 & 1 & & 85_{10} \\
\hline
 & \underline{1} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & & 160_{10}
\end{array}$$

CF	=	0	There is no carry out from MSB
PF	=	1	The result has even number of 1
AC	=	1	There is a carry from bit 3 to bit 4
ZF	=	0	Result is not zero
SF	=	1	Sign bit is 1
OF	=	1	Out of the range (-128_{10} to 127_{10})

+ Example – Intel 8086 (Register Organizations) (15)

- Another 3 bits are known as Control Flags: direction flag (DF), interrupt enable flag (IF) and trap flag
- **Trap flag (TF)**: If TF is set (1), the 8088/8086 goes into single-step mode of operation (for program debugging)
- **Interrupt flag (IF)**: If IF is set (1), the maskable interrupt is enabled. Otherwise, the maskable interrupt is disabled
- **Direction flag (DF)**: This flag determines the direction of string operations; if DF is set (1), string data is transferred from high address to low address; otherwise, the string data is transferred from low address to high address

+ Summary

Lecture B - 02

Structure and Function of CPU, Instruction Cycle

- Processor organization
- Register organization
  - User-visible registers
  - Control and status registers
- Instruction cycle
  - The indirect cycle
  - Data flow
- The 8086 processor family
  - Register organization

+

- Slides adopted from:
  - Computer Organization and Architecture, 9th Edition

    William Stallings

    ISBN-10: 013293633X | ISBN-13: 9780132936330