MIT-WPU

Department of Electrical and Computer Engineering

T.Y. B. Tech (Semester V)

Microcontroller and Applications

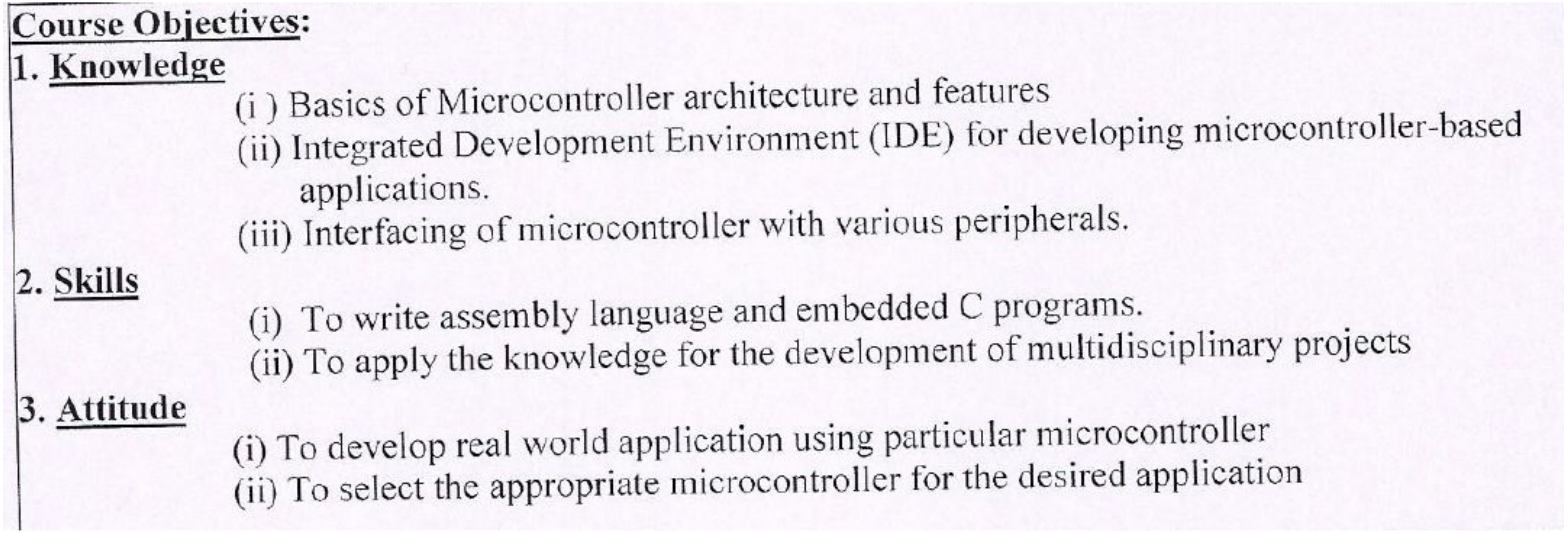
ECE2003B

Course Content

1. Introduction to Microcontroller
2. CIP-51 Architecture
3. Peripheral Interfacing and Programming-I
4. Peripheral Interfacing and Programming-II

Pre-requisites: Digital Electronics

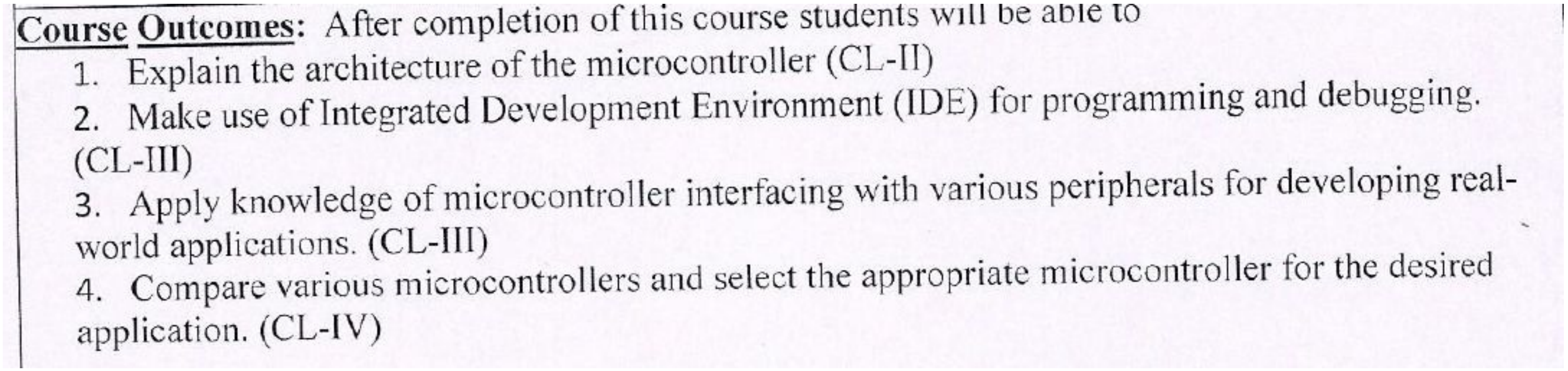
Course Objectives



Course Objectives:

1. **Knowledge**
   - (i) Basics of Microcontroller architecture and features
   - (ii) Integrated Development Environment (IDE) for developing microcontroller-based applications.
   - (iii) Interfacing of microcontroller with various peripherals.
2. **Skills**
   - (i) To write assembly language and embedded C programs.
   - (ii) To apply the knowledge for the development of multidisciplinary projects
3. **Attitude**
   - (i) To develop real world application using particular microcontroller
   - (ii) To select the appropriate microcontroller for the desired application

Course Outcomes



Course Outcomes: After completion of this course students will be able to

1. Explain the architecture of the microcontroller (CL-II)
2. Make use of Integrated Development Environment (IDE) for programming and debugging. (CL-III)
3. Apply knowledge of microcontroller interfacing with various peripherals for developing real-world applications. (CL-III)
4. Compare various microcontrollers and select the appropriate microcontroller for the desired application. (CL-IV)

Syllabus:

- **Introduction to Microcontroller:** Microprocessor and Microcontroller comparison, Microcontroller Architecture comparison, Role of microcontroller in Embedded System, Introduction to CIP-51 architecture and block diagram, Instruction set, and Assembly language programming.
- **CIP-51 Architecture:** Reset sources, Oscillator options, Memory organization, Port structure, Timers, Timer programming, Interrupt handler, Power management modes (All programs in Embedded C).
- **Peripheral Interfacing and Programming-I:** Interfacing of LED, Relay, Buzzer, Switch, 7-segment display, LCD, Keypad, Stepper Motor, DAC, ADC Programming, Programmable Counter Array (PCA), DC motor control using PWM (All programs in Embedded C).
- **Peripheral Interfacing and Programming-II:** Basics of Serial Communication protocol: UART, Study of RS232, RS485, I2C, SPI, EEPROM with SPI (All programs in Embedded C), Comparative study of various emerging microcontrollers, Microcontroller Applications: Case Study.

List of Practical:

1. **Simple assembly language programming.**
2. **Complex assembly language programming.**
3. **Interfacing LED, Relay, Buzzer, and switch with C8051F340.**
4. **Interfacing LCD with C8051F340.**
5. **Interfacing DAC with C8051F340.**
6. **Interfacing ADC with C8051F340.**
7. **Interfacing DC motor and control its speed using PWM with C8051F340.**
8. **Interfacing UART with C8051F340.**
9. **Interfacing Stepper Motor with C8051F340.**
10. **Interfacing EEPROM using SPI with C8051F340.**
11. **Project Based Learning - Design and implement a microcontroller-based project.**

UNIT-I

Introduction to Microcontroller

Contents

Common Terminologies:

- Integrated Circuit (IC): A miniaturized electronic circuit that consists of semiconductor devices and passive components contained in a package.

- Central Processing Unit (CPU): This refers to the core of the MCU that executes code.

- Microcontroller Unit (MCU): This is the standard acronym used for Microcontrollers, and refers to the full IC that contains the CPU and peripherals.

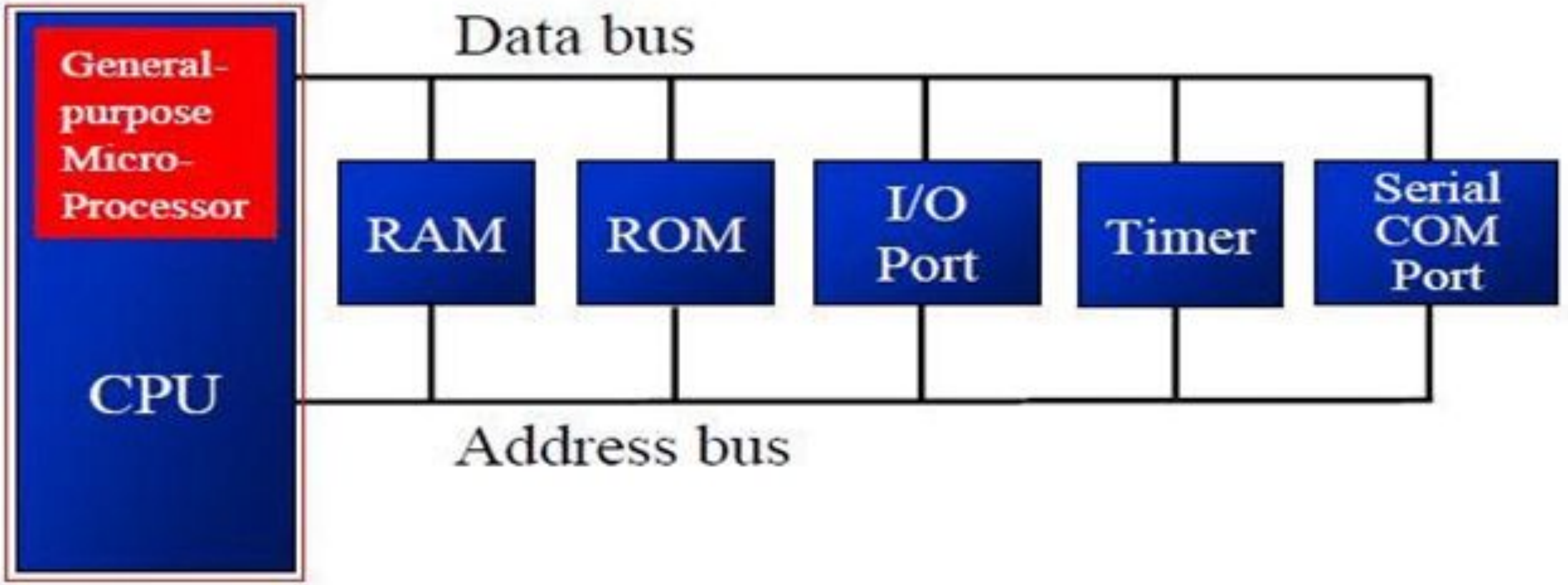
- “n-bit” – the “n” refers to the data bus width of the CPU, and is the maximum width of data it can handle at a time.
  - Examples: 8-bit MCU, 32-bit MCU

Microprocessor and Microcontroller

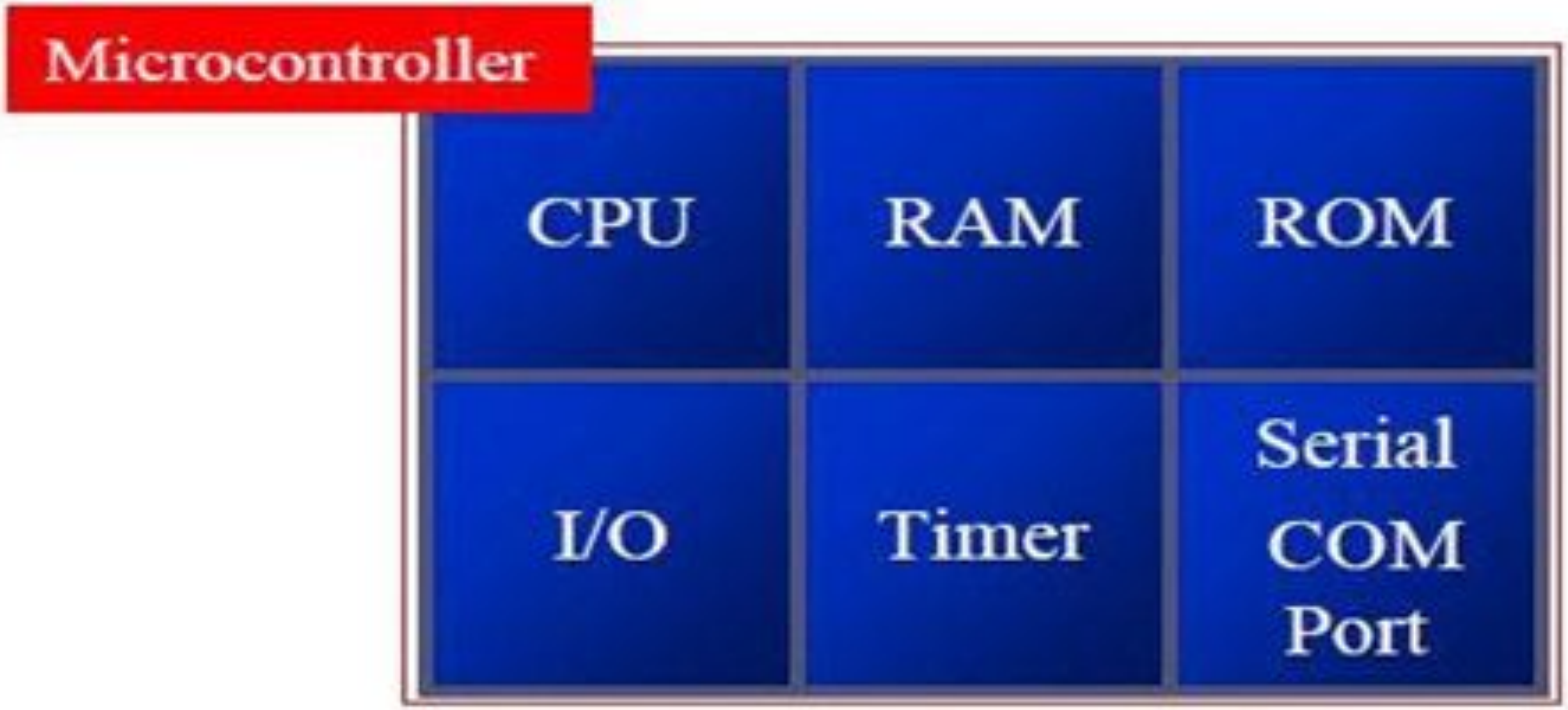
- Microprocessor: General-purpose CPU/ General-purpose Processor (GPP)
  - Emphasis is on flexibility and performance
  - Generic user-interface such as keyboard, mouse, etc.
  - Used in a PC, PDA, cell phone, etc.

- Microcontroller: (Microprocessor + Memory + Other Peripherals) on a single chip
  - Emphasis is on size and cost reduction
  - The user interface is tailored to the application, such as the buttons on a TV remote control
  - Used in a digital watch, TV remote control, car and many common day-to-day appliances

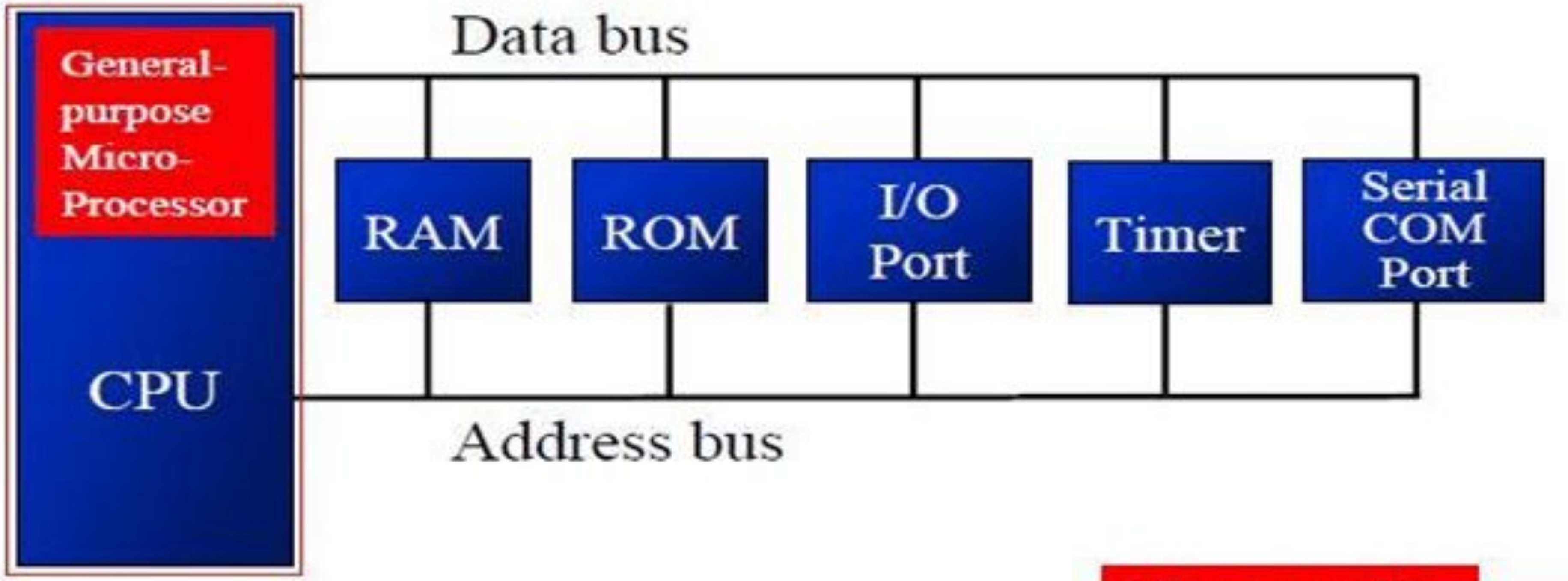
Microprocessor

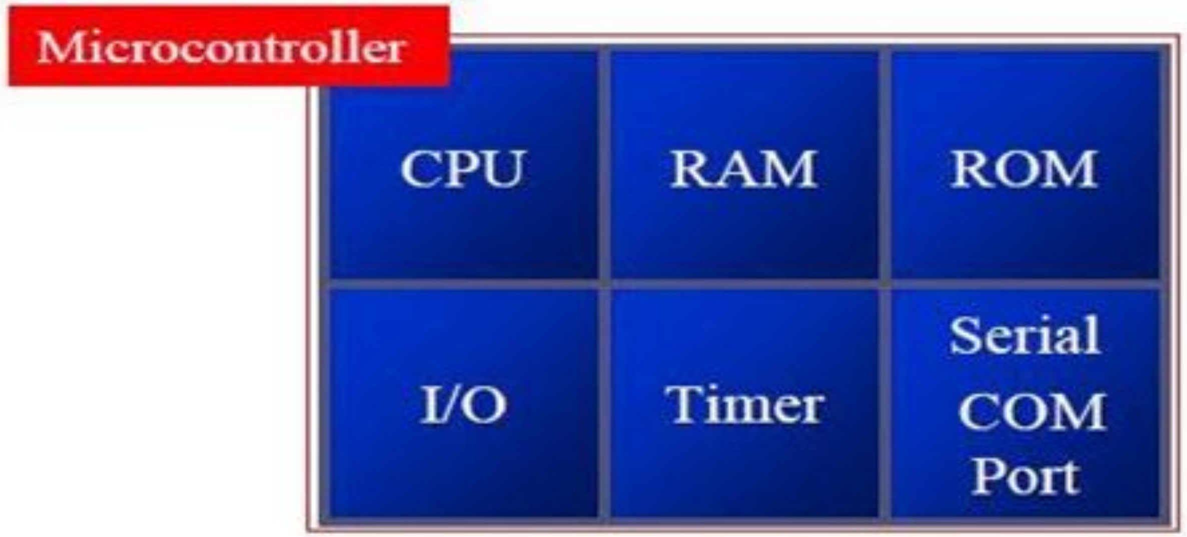


Microcontroller

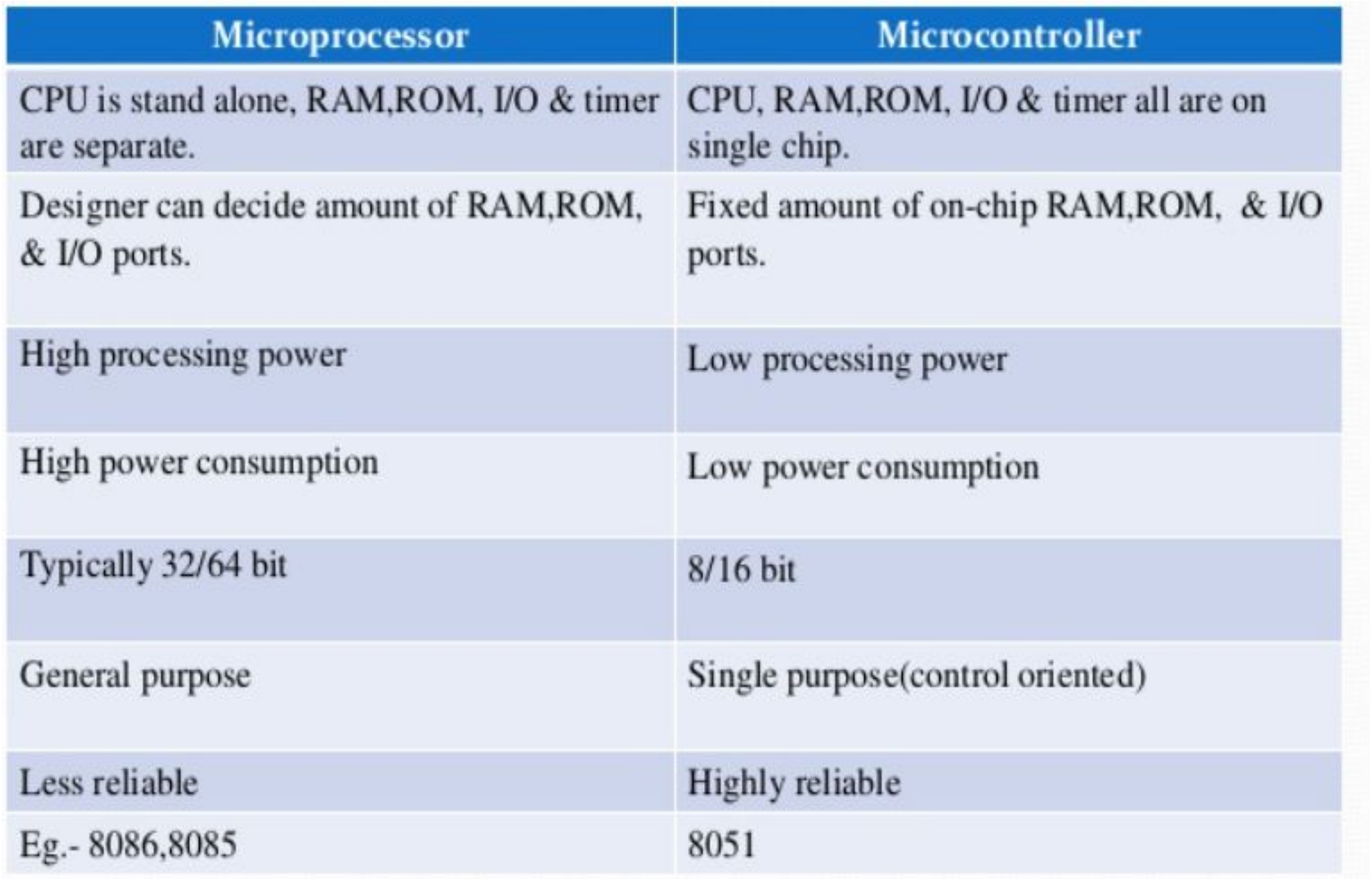


Microprocessor Vs Microcontroller





Microprocessor Vs Microcontroller



Microprocessor	Microcontroller
CPU is stand alone, RAM,ROM, I/O & timer are separate.	CPU, RAM,ROM, I/O & timer all are on single chip.
Designer can decide amount of RAM,ROM, & I/O ports.	Fixed amount of on-chip RAM,ROM, & I/O ports.
High processing power	Low processing power
High power consumption	Low power consumption
Typically 32/64 bit	8/16 bit
General purpose	Single purpose(control oriented)
Less reliable	Highly reliable
Eg.- 8086,8085	8051

Role of microcontroller in Embedded System

- A combination of hardware and software designed to perform a dedicated function
- Embedded systems are computing systems with tightly coupled hardware and software integration.
- Designed to perform dedicated function
- Embedded systems are part of a larger system or product

  -e.g., antilock braking system in a car
- Embedded systems are tightly coupled to their environment imposes real-time constraints by the need to interact with the environment

Role of microcontroller in Embedded System

- A microcontroller is used in an embedded system because it is tasked with processing the flow of data that comes in and goes out.
- It makes decisions according to code which are written by a developer.
- It performs operations based on data it obtains via inputs and sends data telling outputs what actions to perform.
- It also stores information onboard its inbuilt memory.

https://electronicguidebook.com/why-a-microcontroller-is-used-in-an-embedded-system/

Embedded Products Using Microcontrollers (Applications)

- Home
  - Appliances, intercom, telephones, security systems, garage door openers, answering machines, fax machines, home computers, TVs, cable TV tuner, VCR, camcorder, remote controls, video games, cellular phones, musical instruments, sewing machines, lighting control, paging, camera, pinball machines, toys, exercise equipment
- Office
  - Telephones, computers, security systems, fax machines, microwave, copier, laser printer, color printer, paging
- Auto
  - Trip computer, engine control, air bag, ABS, instrumentation, security system, transmission control, entertainment, climate control, cellular phone, keyless entry

Is 8-bit Still Relevant?

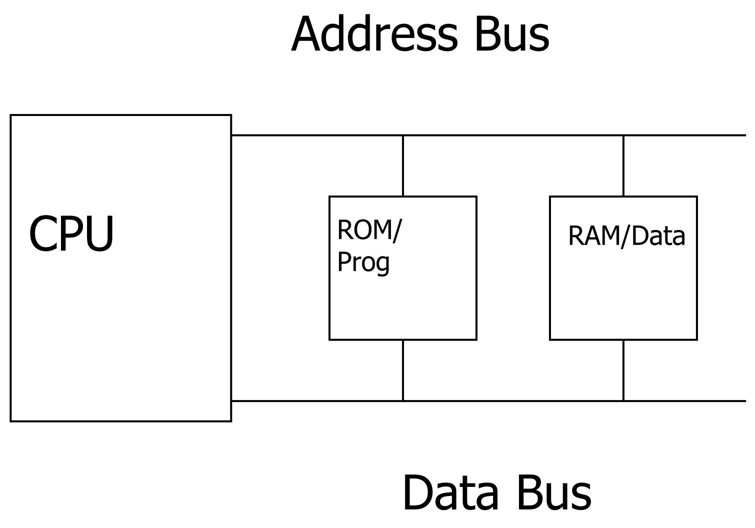
- "n-bit" – the "n" refers to the data bus width of the CPU, and is the maximum width of data it can handle at a time
- 8-bit microcontrollers are sufficient and cost-effective for many embedded applications
- More and more advanced features and peripherals are added to 8-bit processors by various vendors
- 8-bit MCUs are well-suited for low-power applications that use batteries

The 8051 Microcontroller—A Brief History

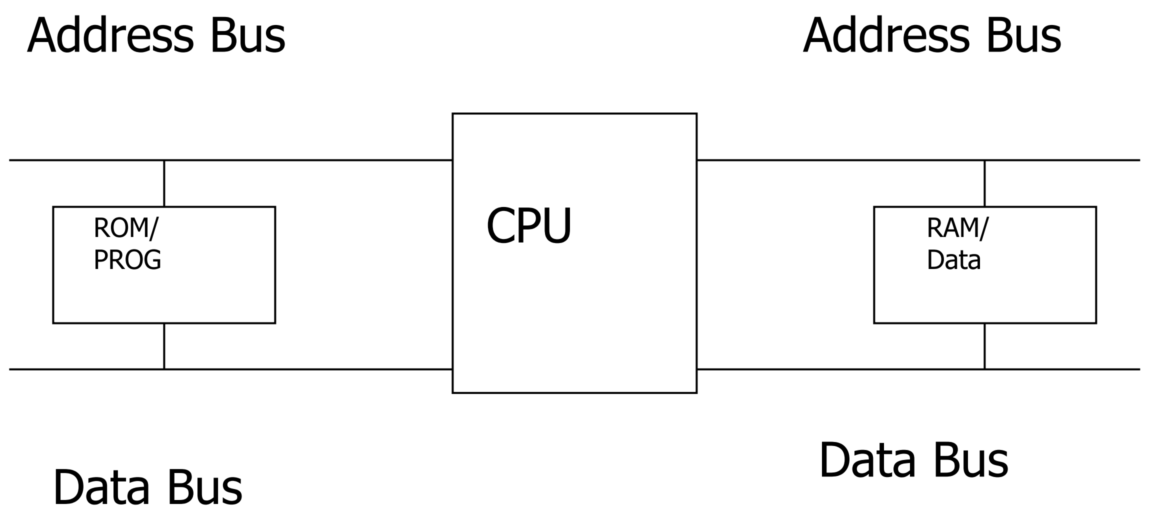
- In 1980, Intel introduced the 8051, relevant today after more than four decades.
- First device in the MCS-51® family of 8-bit microcontrollers.
- In addition to Intel there are other second source suppliers of the ICs, who make microcontrollers that are compatible with the 8051 architecture.
- In recent years some companies have incorporated many different and additional features into 805.
- In 2000, Silicon Laboratories introduced CIP-51 microcontroller chip (C8051F340) based on the 8051 core CPU.
    - This will be the platform for this course.

Von Neumann and Harvard architecture

- Time Division multiplexing in Von-Neumann
- In Harvard fast execution of instructions



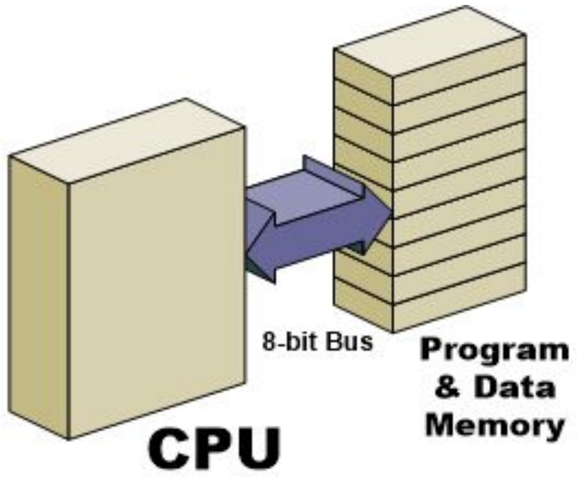
Von Neumann



Harvard

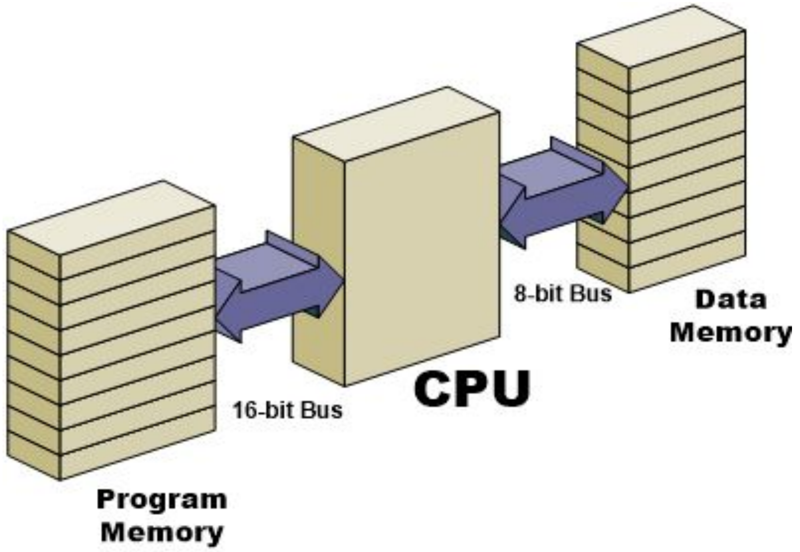
Comparison of Von Neumann and Harvard

Von Neumann architecture

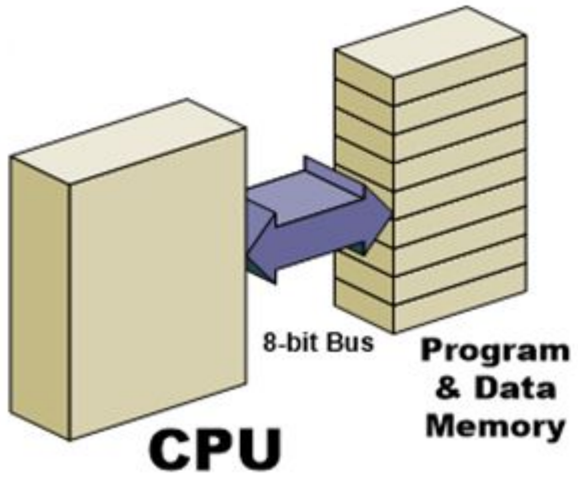
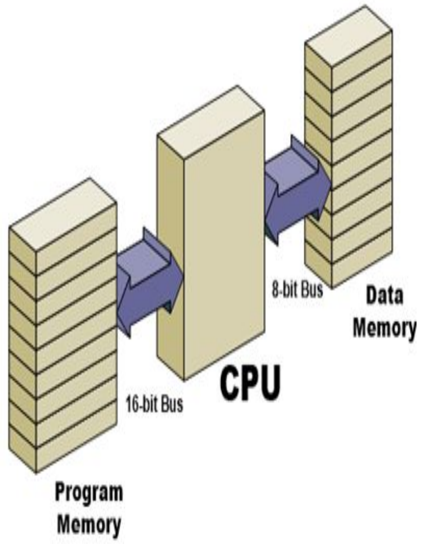


- Fetches instructions and data from a single memory space
- Limits operating bandwidth
- Allows for fixed bus widths
- Architecture --- CISC
- Variable instruction format

Harvard architecture



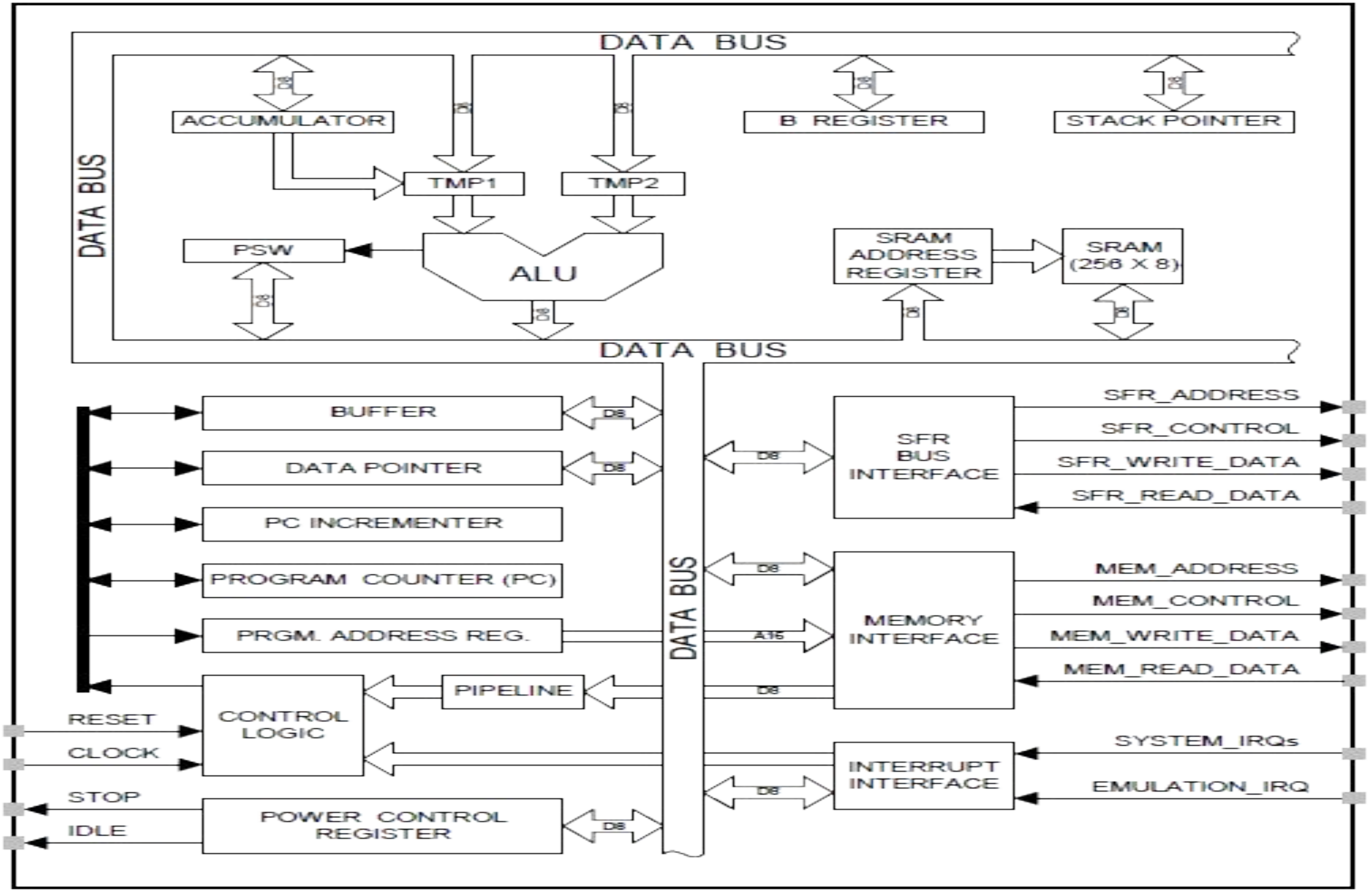
- Fetches Instruction and Data from two separate memory spaces
- Improved operating bandwidth
- Allows for different bus widths
- Architecture --- RISC
- Fixed instruction format

Aspect	**Von Neumann Architecture**	**Harvard Architecture**
		 8-bit Bus Data Memory CPU 16-bit Bus Program Memory
Memory	Same set of address and data bus to access program and data memory	Separate address and data bus to access program and data memory
Performance	Slow in performance because the CPU cannot access both instruction and data memory simultaneously	Improved performance because the CPU can access both instruction and data memory simultaneously
Architecture	CISC	RISC
Applications	Suitable for general-purpose computing where flexibility is required	Suitable for embedded systems where performance is critical and code is not frequently modified

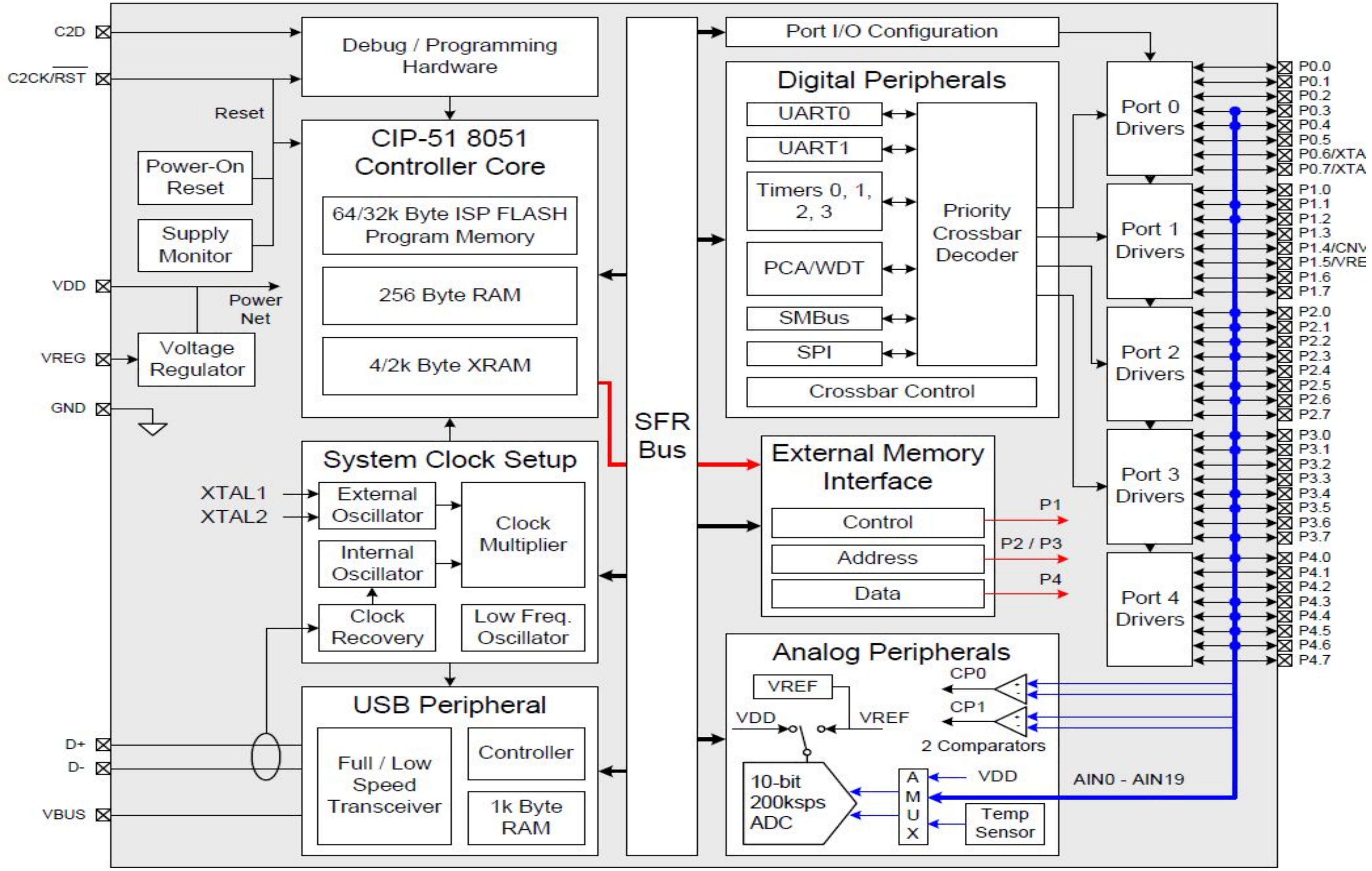
RISC vs. CISC

RISC (Reduced Instruction Set Computer)	CISC (Complex Instruction Set Computer)
1. Simple Instruction taking 1 cycle	1. Complex Instruction taking multiple cycles
2. Only LOADs, STOREs access memory	2. Any Instruction may access memory
3. Designed around pipeline	3. Designed around Instruction Set
4. Instruction executed by h/w	4. Instruction interpreted by micro program
5. Fixed format Instruction	5. Variable format Instruction
6. Few Instruction and modes	6. Many Instruction and modes
7. Complexity in the compiler	7. Complexity in the micro program
8. Multiple register sets	8. Single register set
9. Operates at 50-150 MHz	9. Operates at 33-50 MHz

Introduction to CIP-51 architecture



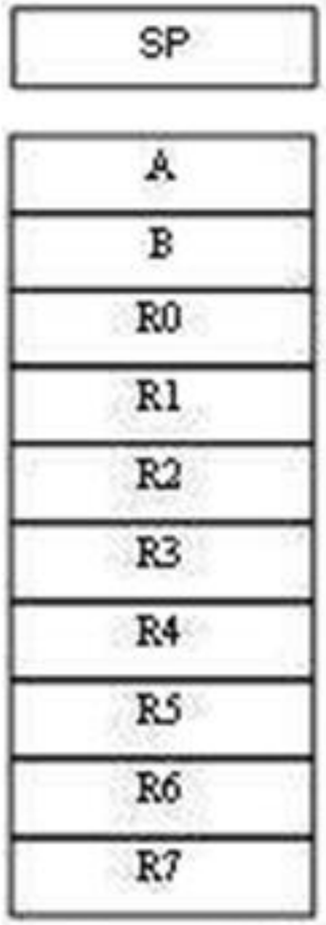
Block Diagram C8051F340



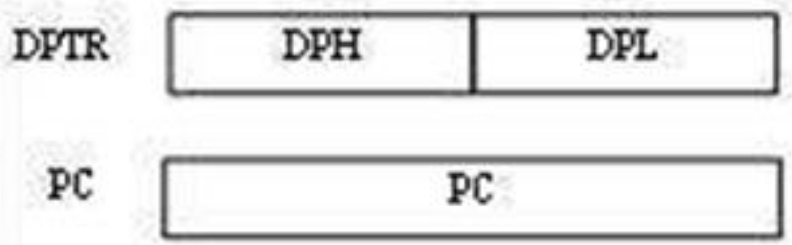
C8051F340 Highlighted Features

- High-speed pipelined 8051-compatible microcontroller core (up to 48 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- Universal Serial Bus (USB) Function Controller
- True 10-bit 200 ksps differential / single-ended ADC with analog multiplexer
- On-chip Voltage Reference and Temperature Sensor
- On-chip Voltage Comparators (2)
- Precision internal calibrated 12 MHz internal oscillator and 4x clock multiplier
- Internal low-frequency oscillator for additional power savings
- Up to 64 kB of on-chip Flash memory
- Up to 4352 Bytes of on-chip RAM (256 + 4 kB)
- External Memory Interface (EMIF) available on 48-pin versions.
- SMBus/I2C, up to 2 UARTs, and Enhanced SPI serial interfaces implemented in hardware
- Four general-purpose 16-bit timers
- Programmable Counter/Timer Array (PCA) with five capture/compare modules and Watchdog Timer function
- On-chip Power-On Reset, VDD Monitor, and Missing Clock Detector
- Up to 40 Port I/O (5 Ports)

Registers supported by CIP-51 Core

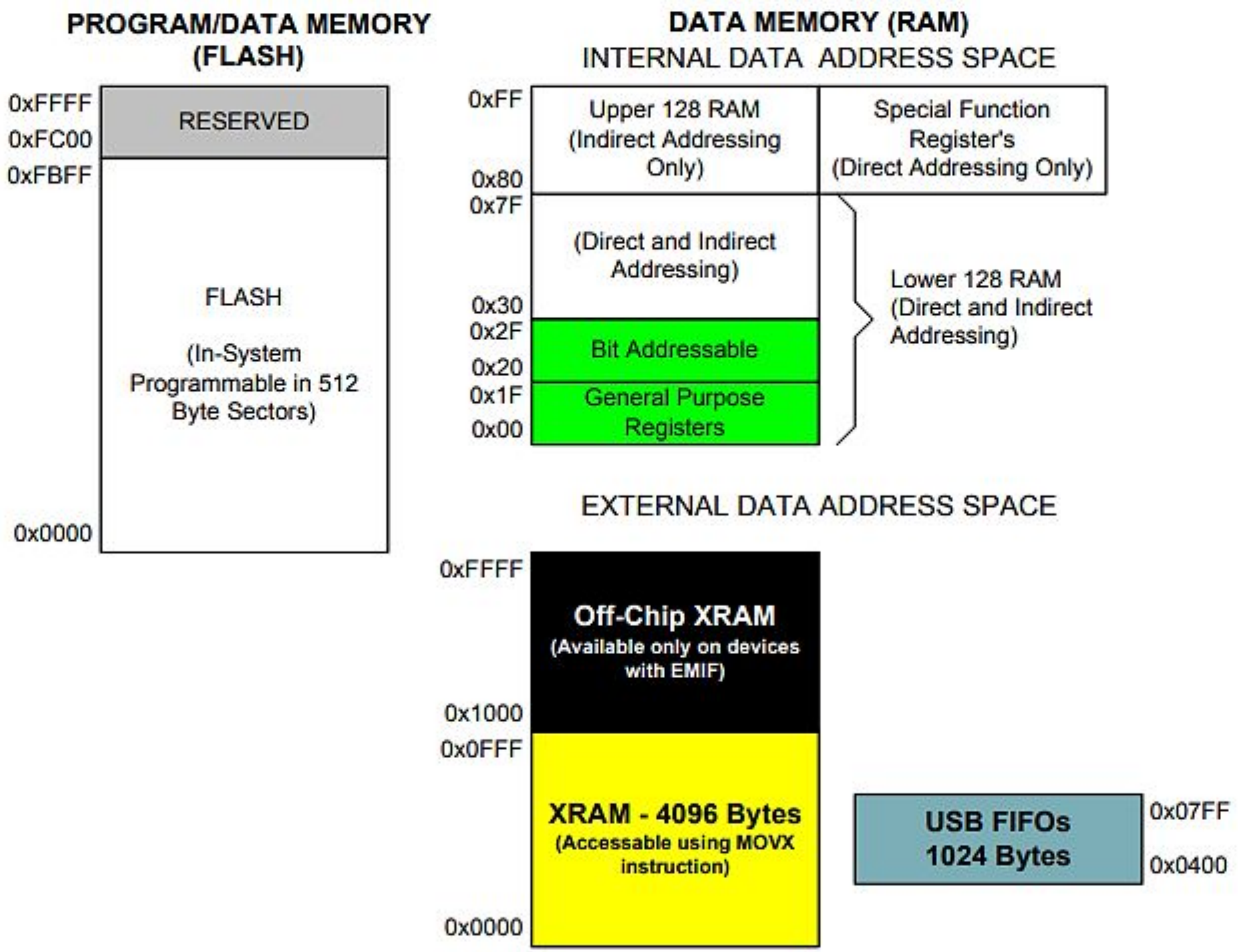


8-bit Registers of CIP-51



16-bit Registers of CIP-51

Memory Organization



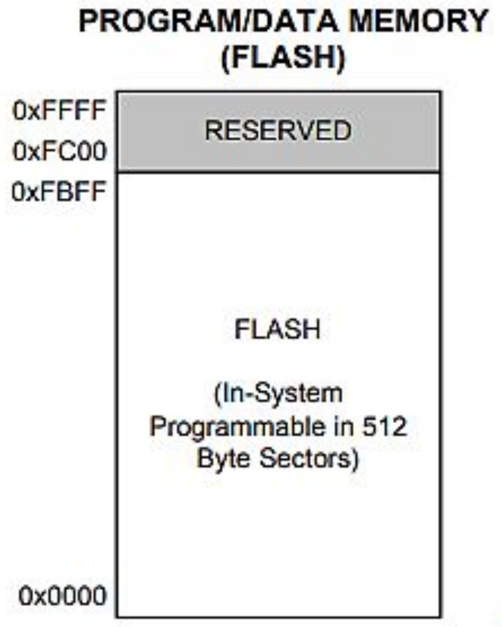
PROGRAM/DATA MEMORY (FLASH)
0xFFFF
0xFC00
RESERVED
0xFBFF
FLASH
(In-System Programmable in 512 Byte Sectors)
0x0000
DATA MEMORY (RAM)
INTERNAL DATA ADDRESS SPACE
0xFF
Upper 128 RAM (Indirect Addressing Only)
Special Function Register's (Direct Addressing Only)
0x80
0x7F
(Direct and Indirect Addressing)
0x30
0x2F
Bit Addressable
0x20
0x1F
General Purpose Registers
0x00
Lower 128 RAM (Direct and Indirect Addressing)
EXTERNAL DATA ADDRESS SPACE
0xFFFF
Off-Chip XRAM
(Available only on devices with EMIF)
0x1000
0x0FFF
XRAM - 4096 Bytes
(Accessable using MOVX instruction)
0x0000
USB FIFOs
1024 Bytes
0x07FF
0x0400

Figure 9.2. On-Chip Memory Map for 64 kB Devices

- The memory organization of C8051F340 is similar to that of a standard 8051
- Program and data memory share the same address space but are accessed via different instruction types

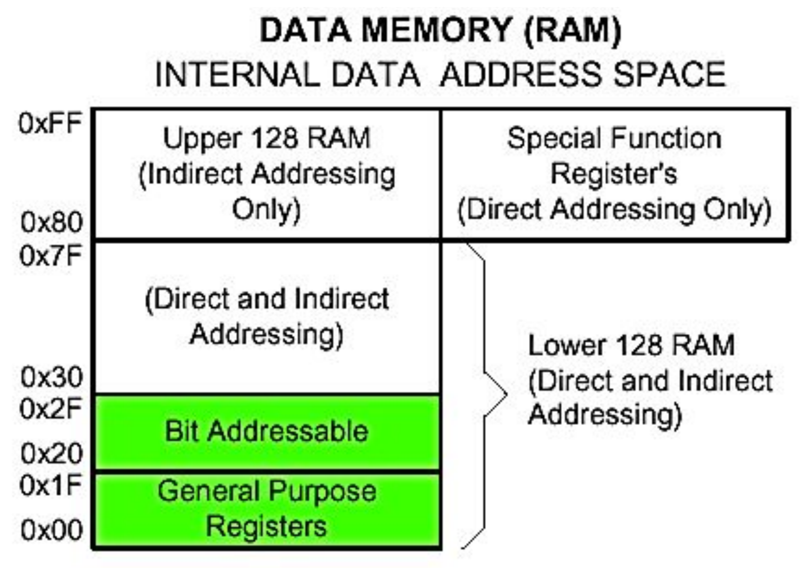
Program Memory

- FLASH memory
  - Can be reprogrammed in-circuit
  - Provides non-volatile data storage
  - Allows field upgrades of the 8051 firmware
- The C8051F340's program memory consists of 65536 bytes of FLASH
- 1024 bytes at address 0xFC00 to 0xFFFF are reserved


PROGRAM/DATA MEMORY (FLASH)
0xFFFF
0xFC00
RESERVED
0xFBFF
FLASH
(In-System Programmable in 512 Byte Sectors)
0x0000

Internal Data Memory

- The internal data memory consists of 256 bytes of RAM
- The special function registers (SFR) are accessed when the direct addressing mode is used to access the upper 128 bytes of memory locations from 0x80 to 0xFF
- The general purpose RAM are accessed when indirect addressing is used to access the upper 128 bytes
- The first 32 bytes of the internal data memory are addressable as four banks of 8 general purpose registers
- The next 16 bytes are bit-addressable or byte-addressable


DATA MEMORY (RAM)
INTERNAL DATA ADDRESS SPACE
0xFF
0x80
0x7F
0x30
0x2F
0x20
0x1F
0x00
Upper 128 RAM (Indirect Addressing Only)
Special Function Register's (Direct Addressing Only)
(Direct and Indirect Addressing)
Bit Addressable
General Purpose Registers
Lower 128 RAM (Direct and Indirect Addressing)

Special Function Registers

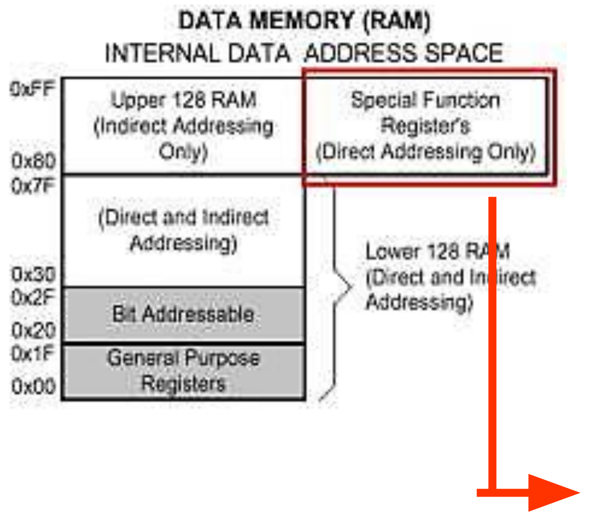
- SFRs provide control and data exchange with the microcontroller’s resources and peripherals

- The C8051F020 duplicates the SFRs found in a typical 8051 implementation
  - The C8051F020 implements additional SFRs which are used to configure and access the sub-systems unique to the microcontroller

- This allows the addition of new functionalities while retaining compatibility with the MCS-51™ instruction set

- The SFRs with addresses ending in 0x0 or 0x8 (e.g. P0, TCON, P1, SCON, IE, etc.) are bit-addressable as well as byte-addressable

Special Function Registers


DATA MEMORY (RAM)
INTERNAL DATA ADDRESS SPACE
0xFF
0x80
0x7F
0x30
0x2F
0x20
0x1F
0x00
Upper 128 RAM (Indirect Addressing Only)
Special Function Register's (Direct Addressing Only)
(Direct and Indirect Addressing)
Bit Addressable
General Purpose Registers
Lower 128 RAM (Direct and Indirect Addressing)

	0(8) Bit addressable	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)
F8	SPI0CN	PCA0H	PCA0CPH0	PCA0CPH1	PCA0CPH2	PCA0CPH3	PCA0CPH4	WDTCN
F0	B	SCON1	SBUF1	SADDR1	TL4	TH4	EIP1	EIP2
E8	ADC0CN	PCA0L	PCA0CPL0	PCA0CPL1	PCA0CPL2	PCA0CPL3	PCA0CPL4	RSTSRC
E0	ACC	XBR0	XBR1	XBR2	RCAP4L	RCAP4H	EIE1	EIE2
D8	PCA0CN	PCA0MD	PCA0M0	PCA0CPM1	PCA0CPM2	PCA0CPM3	PCA0CPM4	
D0	PSW	REF0CN	DAC0L	DAC0H	DAC0CN	DAC1L	DAC1H	DAC1CN
C8	T2CON	T4CON	RCAP2L	RCAP2H	TL2	TH2		SMB0CR
C0	SMB0CN	SMB0STA	SMB0DAT	SMB0ADR	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH
B8	IP	SADEN0	AMX0CF	AMX0SL	ADC0CF	P1MDIN	ADC0L	ADC0H
B0	P3	OSCXCN	OSCICN			P74OUT	FLSCL	FLACL
A8	IE	SADDR0	ADC1CN	ADC1CF	AMX1SL	P3IF	SADEN1	EMI0CN
A0	P2	EMI0TC		EMI0CF	P0MDOUT	P1MDOUT	P2MDOUT	P3MDOUT
98	SCON0	SBUF0	SPI0CFG	SPI0DAT	ADC1	SPI0CKR	CPT0CN	CPT1CN
90	P1	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H	P7	
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PSCTL
80	P0	SP	DPL	DPH	P4	P5	P6	PCON

Instruction Set and Assembly Programming

- An assembly language instruction has four fields:
- [Label:] mnemonic [operands] [;comments]
- Eg. MOV A, #00H   ;put 0 in the accumulator
  - A = 00000000
- An instruction is made up of an operation code (op-code) followed by either zero, one or two bytes of operands
- The op-code identifies the type of operation to be performed while the operands identify the source and destination of the data
- The operand can be:
  - The data value itself
  - A CPU register
  - A memory location
  - An I/O port

Addressing Modes

- Five modes of addressing
- The different addressing modes determine how the operand byte is selected

Addressing Modes	Instruction
Register	MOV A, B
Direct	MOV 30H,A
Indirect	ADD A,@R0
Immediate	ADD A,#80H
Indexed	MOVC A,@A+PC

Arithmetic Flags

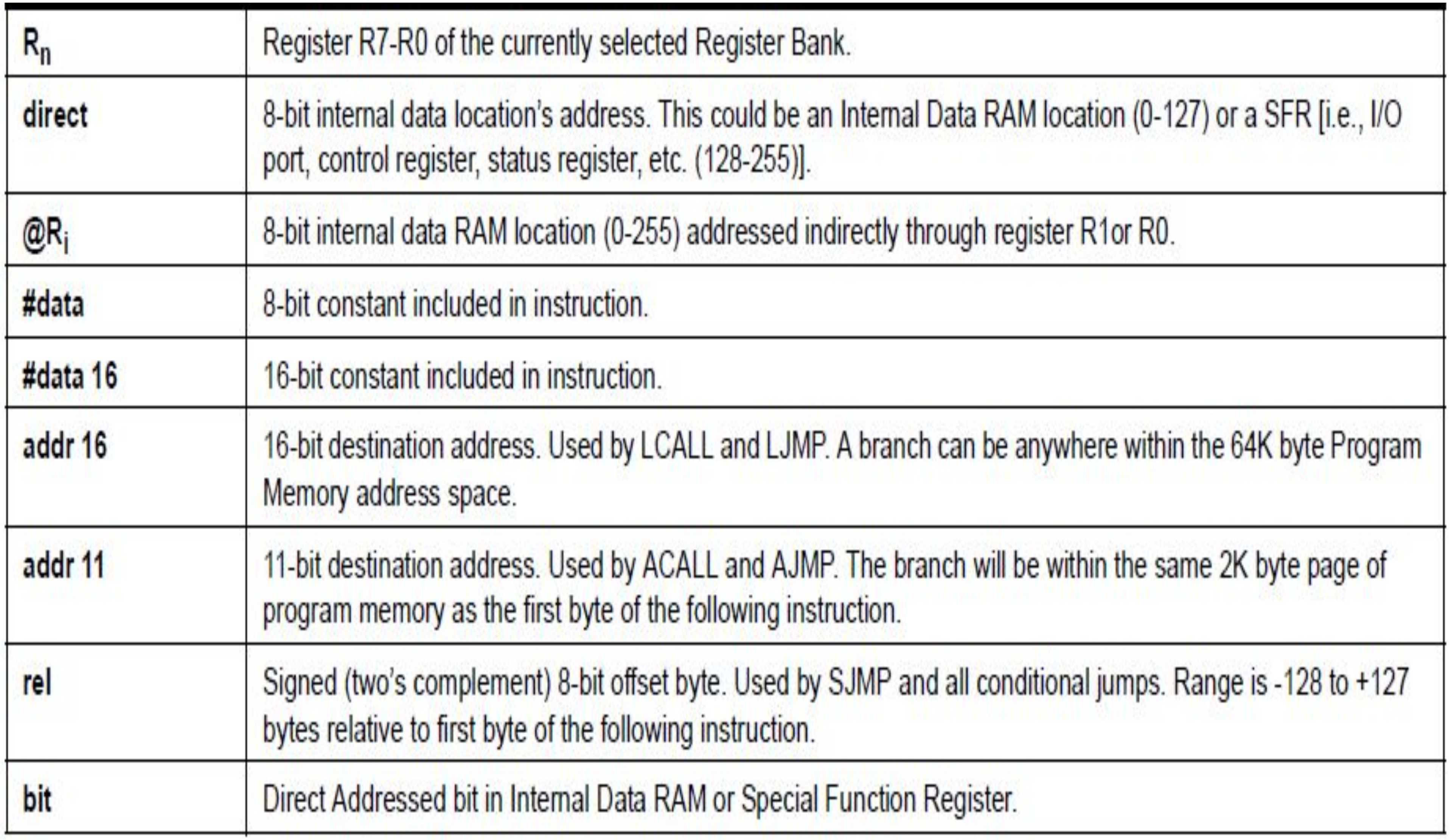
- **Flag**: It is a 1-bit register that indicates the status of the result from an operation
- Flags are either at a flag-state of value **0** or **1**
- Arithmetic flags indicate the status of the results from mathematical operations (+, −, *, /)
- There are 4 arithmetic flags in the 8051
  - Carry (C)
  - Auxiliary Carry (AC)
  - Overflow (OV)
  - Parity (P)

Program Status Word (PSW)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	AC	FO	RS1	RS0	OV	UD	P

Symbol	Function
CY	**Carry flag**
AC	**Auxiliary Carry flag (For BCD Operations)**
F0	**Flag 0 (Available to the user for General Purpose)**
RS1, RS0	**Register bank select: RS1 RS0 Working Register Bank and Address 0 0 Bank0 0 1 Bank1 1 0 Bank2 (D:0x10 - D:0x17) 1 1 Bank3 (D:0x18H - D:0x1F)**
0V	**Overflow flag**
UD	**User definable flag**
P	**Parity flag; P=1 for Odd no. of 1’s; P=0 for Even no. of 1’s**

Related Terms



R_n	Register R7-R0 of the currently selected Register Bank.
direct	8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].
$@R_i$	8-bit internal data RAM location (0-255) addressed indirectly through register R1or R0.
#data	8-bit constant included in instruction.
#data 16	16-bit constant included in instruction.
addr 16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K byte Program Memory address space.
addr 11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
bit	Direct Addressed bit in Internal Data RAM or Special Function Register.

MOV dest, source dest □ source

Sr. No	Mnemonic	Description
1	MOV A, Rn	Move register to accumulator
2	MOV A, direct	Move direct byte to accumulator
3	MOV A, @ Ri	Move indirect RAM to accumulator
4	MOV A, #data	Move immediate data to accumulator
5	MOV Rn, A	Move accumulator to register
6	MOV Rn, direct	Move direct byte to register
7	MOV Rn, #data	Move immediate data to register
8	MOV direct, A	Move accumulator to direct byte
9	MOV direct, Rn	Move register to direct byte
10	MOV direct, direct	Move direct byte to direct
11	MOV direct, @Ri	Move indirect RAM to direct byte
12	MOV direct, #data	Move immediate data to direct byte
13	MOV @Ri, A	Move accumulator to indirect RAM
14	MOV @Ri, direct	Move direct byte to indirect RAM
15	MOV @Ri, #data	Move immediate data to indirect RAM

Data Transfer Instructions

16	MOV DPTR, #data 16	Load data pointer with a 16-bit constant
17	MOVC A, @A+DPTR	Move code byte relative to DPTR to accumulator
18	MOVC A,@A+PC	Move code byte relative to PC to accumulator
19	MOVX A, @Ri	Move external RAM (8-bit addr) to accumulator
20	MOVX A, @DPTR	Move external RAM (16-bit addr) to accumulator
21	MOVX @Ri, A	Move accumulator to external RAM (8-bit addr)
22	MOVX @DPTR,A	Move accumulator to external RAM (16-bit addr)
23	PUSH direct	Push direct byte onto stack
24	POP direct	Pop direct byte from stack
25	XCH A, Rn	Exchange register byte with accumulator
26	XCH a, direct	Exchange direct byte with accumulator
27	XCH A, @Ri	Exchange indirect RAM with accumulator
28	XCHD A, @Ri	Exchange low order digit indirect RAM with accumulator

Arithmetic Instructions

Sr. No	Mnemonic	Description
29	ADD A, Rn	Add register to accumulator
30	ADD A, Direct	Add direct byte to accumulator
31	ADD A, @ Ri	Add indirect RAM to accumulator
32	ADD A, #data	Add immediate data to accumulator
33	ADDC A, Rn	Add register to accumulator with carry
34	ADDC A, direct	Add direct byte to accumulator with carry
35	ADDC A, @Ri	Add indirect RAM to accumulator with carry
36	ADDC A, #data	Add immediate data to accumulator with carry
37	SUBB A, Rn	Subtract register from accumulator with borrow
38	SUBB A, Direct	Subtract direct byte from accumulator with borrow
39	SUBB A, @Ri	Subtract indirect RAM from accumulator with borrow
40	SUBB A, #data	Subtract immediate data from accumulator with borrow

Arithmetic Instructions

41	INC A	Increment Accumulator
42	INC Rn	Increment register
43	INC direct	Increment direct byte
44	INC @Ri	Increment indirect RAM
45	DEC A	Decrement accumulator
46	DEC Rn	Decrement register
47	DEC direct	Decrement direct RAM
48	DEC @Ri	Decrement indirect RAM
49	INC DPTR	Increment data pointer
50	MUL AB	Multiply A and B
51	DIV AB	Divide A by B
52	DA A	Decimal adjust accumulator

Logical Instructions

Sr. No	Mnemonic	Description
53	ANL A, Rn	AND register to accumulator
54	ANL A, Direct	AND direct byte to accumulator
55	ANL A, @ Ri	AND indirect RAM to accumulator
56	ANL A, #data	AND immediate data to accumulator
57	ANL direct, A	AND accumulator to direct byte
58	ANL direct, #data	AND immediate data to direct byte
59	ORL A, Rn	OR register to accumulator
60	ORL A, direct	OR direct byte to accumulator
61	ORL A, @Ri	OR indirect RAM to accumulator
62	ORL A, #Data	OR immediate data to accumulator
63	ORL direct, A	OR accumulator to direct byte

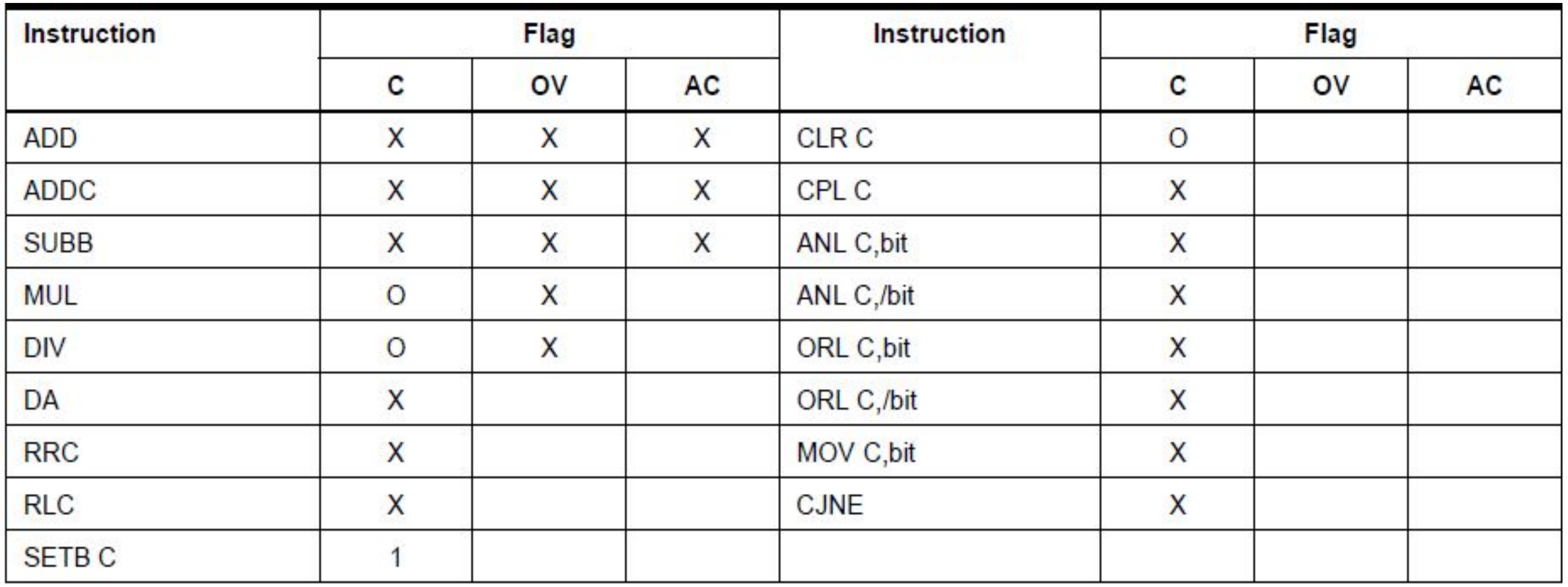
Logical Instructions

64	ORL direct, #data	OR immediate data to direct byte
65	XRL A, Rn	EX-OR register to accumulator
66	XRL A, direct	EX-OR direct byte to accumulator
67	XRL A, @Ri	EX-OR indirect RAM to accumulator
68	XRL A, #Data	EX-OR immediate data to accumulator
69	XRL direct, A	EX-OR accumulator to direct byte
70	XRL direct, #data	EX-OR immediate data to direct byte
71	CLR A	Clear Accumulator
72	CPL A	Complement Accumulator
73	RL A	Rotate accumulator left
74	RLC A	Rotate accumulator left through carry
75	RR A	Rotate accumulator right
76	RRC A	Rotate accumulator right through carry
77	SWAP A	Swap nibbles within the accumulator

Boolean Variable Manipulation Instructions

Sr. No	Mnemonic	Description
78	CLR C	Clear Carry
79	CLR bit	Clear direct bit
80	SETB C	Set Carry
81	SETB bit	Set direct bit
82	CPL C	Complement carry
83	CPL bit	Complement direct bit
84	ANL C, bit	AND direct bit to carry
85	ANL C, /bit	AND complement of direct bit to carry
86	ORL C, bit	OR direct bit to carry
87	ORL C, /bit	OR complement of direct bit to carry
88	MOV C, bit	Move direct bit to carry
89	MOV bit, C	Move carry to direct bit
90	JC rel	Jump if carry is set
91	JNC rel	Jump if carry not set
92	JB bit, rel	Jump if direct bit is set
93	JNB bit, rel	Jump is direct bit is not set
94	JBC bit, rel	Jump if direct bit is set and clear bit

Instructions that affect PSW



Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

X can be 0 or 1

Program Branching Instructions

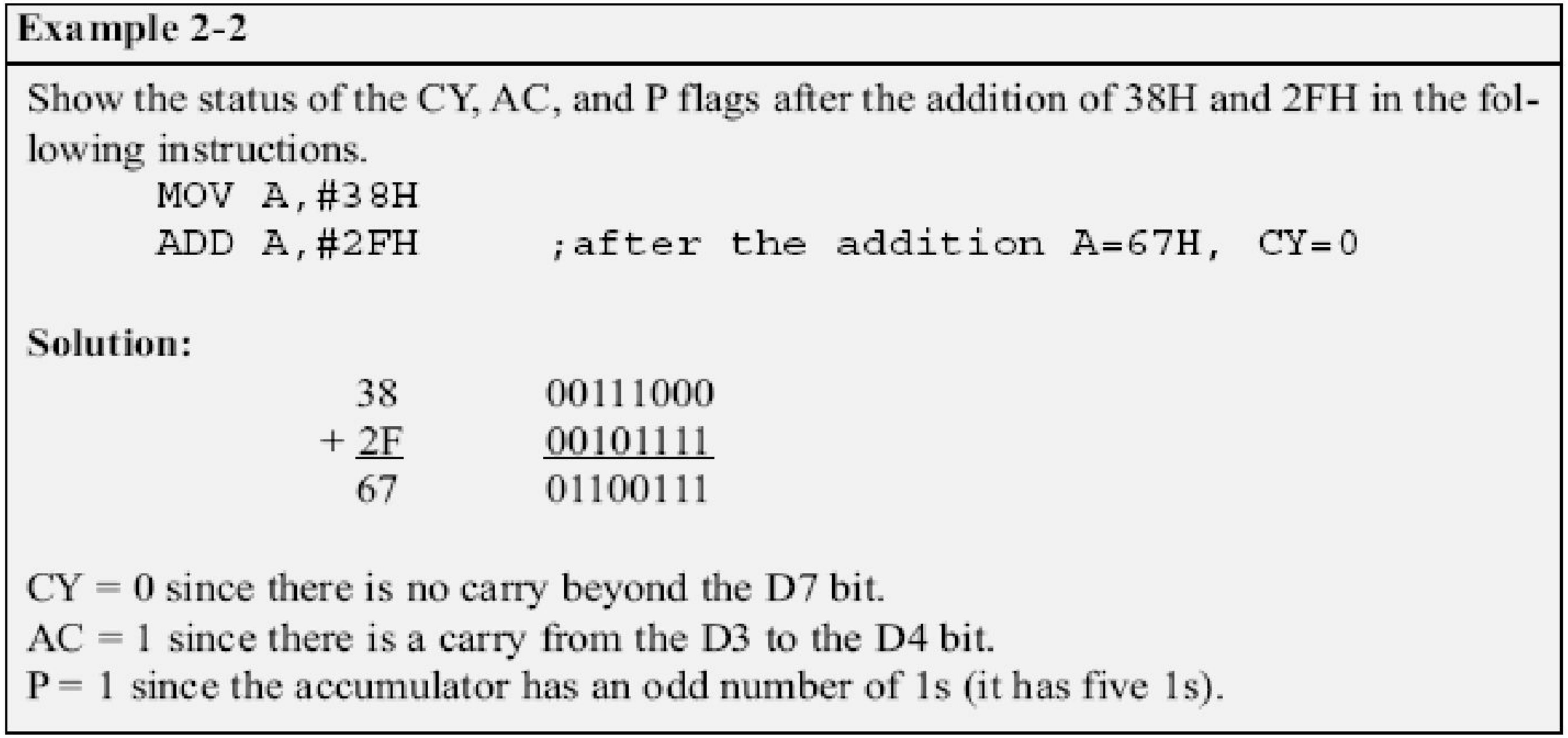
Sr. No	Mnemonic	Description
95	ACALL addr11	Absolute subroutine call
96	LCALL addr16	Long subroutine call
97	RET	Return from subroutine
98	RETI	Return from interrupt
99	AJMP addr11	Absolute Jump
100	LJMP addr16	Long Jump
101	SJMP rel	Short Jump (relative addr)
102	JMP @A+DPTR	Jump indirect relative to DPTR
103	JZ rel	Jump is accumulator is zero
104	JNZ rel	Jump if accumulator is not zero
105	CJNE A, direct, rel	Compare direct byte to accumulator and jump if not equal
106	CJNE A, #data, rel	Compare immediate to accumulator and jump if not equal
107	CJNE Rn, #data, rel	Compare immediate to register and jump if not equal
108	CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal
109	DJNZ Rn, rel	Decrement register and jump if not zero
110	DJNZ direct, rel	Decrement direct byte and jump if not zero
111	NOP	No operation

Assembler Directives and Data Type

- Assembler Directives
  - ORG
  - EQU
  - END
- Data Type:
  - DB

Note: For the mnemonic, number of bytes, and number of clock cycles for each instruction refer Table 9.1 in the datasheet

The flag bits affected by the ADD instruction are CY, P, AC, and OV



Example 2-2

Show the status of the CY, AC, and P flags after the addition of 38H and 2FH in the following instructions.

```
      MOV A,#38H
      ADD A,#2FH      ;after the addition A=67H, CY=0
```

Solution:

```
        38        00111000
      + 2F        00101111
        67        01100111
```

CY = 0 since there is no carry beyond the D7 bit.
AC = 1 since there is a carry from the D3 to the D4 bit.
P = 1 since the accumulator has an odd number of 1s (it has five 1s).

Assembly Programs

- Addition of n 8-bit numbers
- Addition of BCD numbers
- Addition of two 16-bit numbers
- Find square of a number using DPTR
- Program to count number of Odd and Even numbers from a given array
- Program to count number of Positive and Negative numbers from a given array

References

https://www.silabs.com/documents/public/datasheets/C8051F34x.pdf

Muhammas Mazidi, Janice Mazidi and Rolin McKinlay,
“ The 8051 Microcontroller and Embedded Systems using Assembly and C”, Pearson Education, 2nd edition

Instruction set

http://www.keil.com/support/man/docs/is51/is51_xch.htm