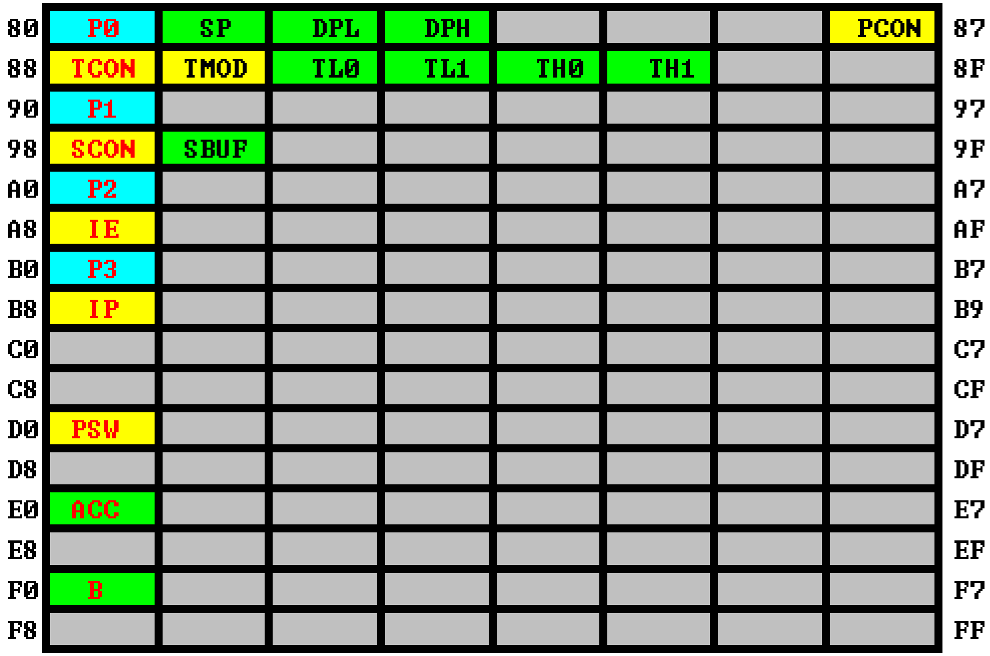
Byte order, special function registers



80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

Overview

- Byte order
  - Little Endian vs. Big Endian
  - Converting to/from decimal
- 8051 Special Function Registers
  - Usage
  - Addressing

Byte order

- Byte order
  - Order of individually addressable values that make up a large data type
  - e.g. 16-bit integer composed of 2 bytes
  - Which byte stored first in memory?
  - Which byte sent first over the network?

IEN 137

Danny Cohen
U S C/I S I
1 April 1980

ON HOLY WARS AND A PLEA FOR PEACE

INTRODUCTION

This is an attempt to stop a war. I hope it is not too late and that somehow, magically perhaps, peace will prevail again.

The latecomers into the arena believe that the issue is: "What is the proper byte order in messages?".

The root of the conflict lies much deeper than that. It is the question of which bit should travel first, the bit from the little end of the word, or the bit from the big end of the word? The followers of the former approach are called the Little-Endians, and the followers of the latter are called the Big-Endians. The details of the holy war between the Little-Endians and the Big-Endians are documented in [6] and described, in brief, in the Appendix. I recommend that you read it at this point.

SWIFT's POINT

It may be interesting to notice that the point which Jonathan Swift tried to convey in Gulliver's Travels in exactly the opposite of the point of this note.

Swift's point is that the difference between breaking the egg at the little-end and breaking it at the big-end is trivial. Therefore, he suggests, that everyone does it in his own preferred way.

We agree that the difference between sending eggs with the little- or the big-end first is trivial, but we insist that everyone must do it in the same way, to avoid anarchy. Since the difference is trivial we may choose either way, but a decision must be made.

Endianness

- Big Endian
  - Most significant byte first
  - Similar to how we write numbers
  - e.g. Sun SPARC, Motorola
- Little Endian
  - Least significant byte first
  - e.g. x86
- 8052?
  - Stay tuned

Endian example

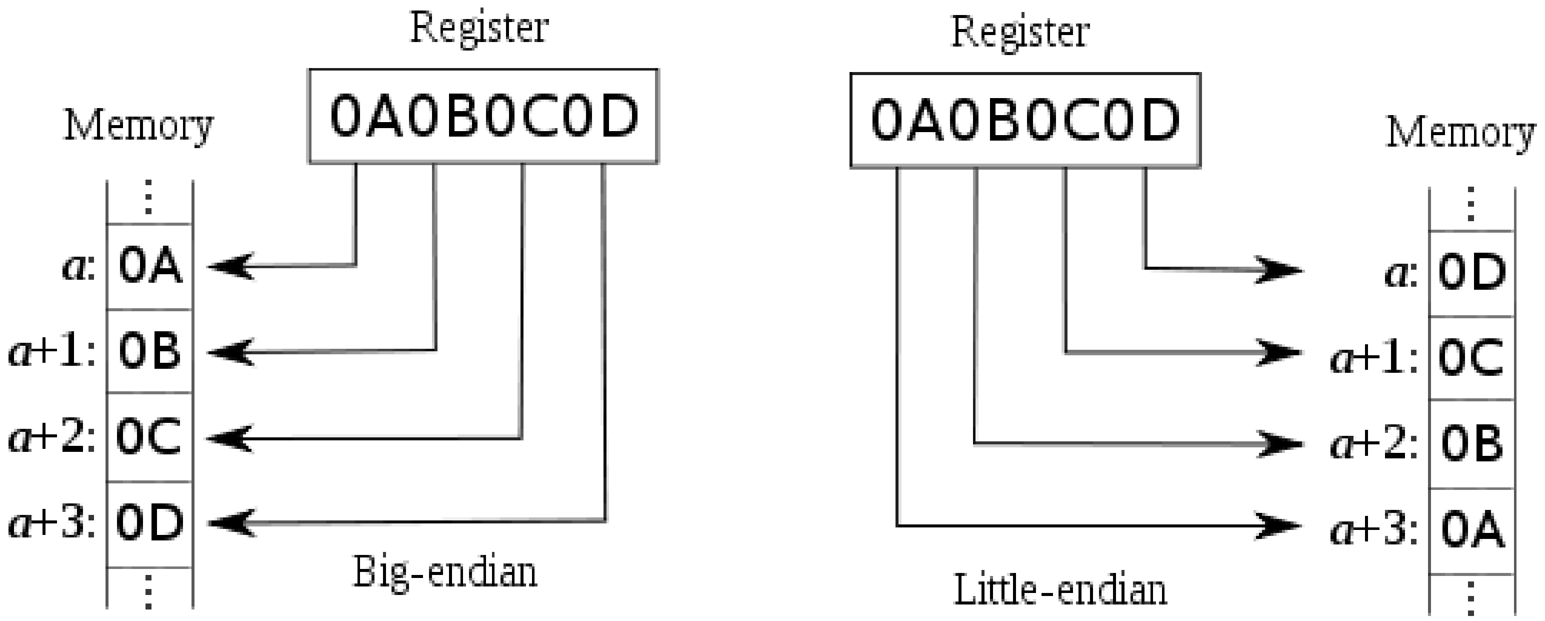
- Example: 32-bit integer in memory
  - Decimal: 272,147,125
  - Hex: 10 38 A2 B5

Memory address	Value
1000h	10
1001h	38
1002h	A2
1003h	B5

Memory address	Value
1000h	B5
1001h	A2
1002h	38
1003h	10

Endian example

- Example: 32-bit register
  - Write into 4 atomic 8-bit memory addresses



Conversion to decimal

- Goal: convert 2 bytes (16-bits) into decimal

Memory address	Hex value
1000h	2B
1001h	1C

Conversion to decimal

- Goal: convert 2 bytes (16-bits) into decimal

Memory address	Hex value
1000h	2B
1001h	1C

2B = $16^1 * 2 + 16^0 * 11 = 43$

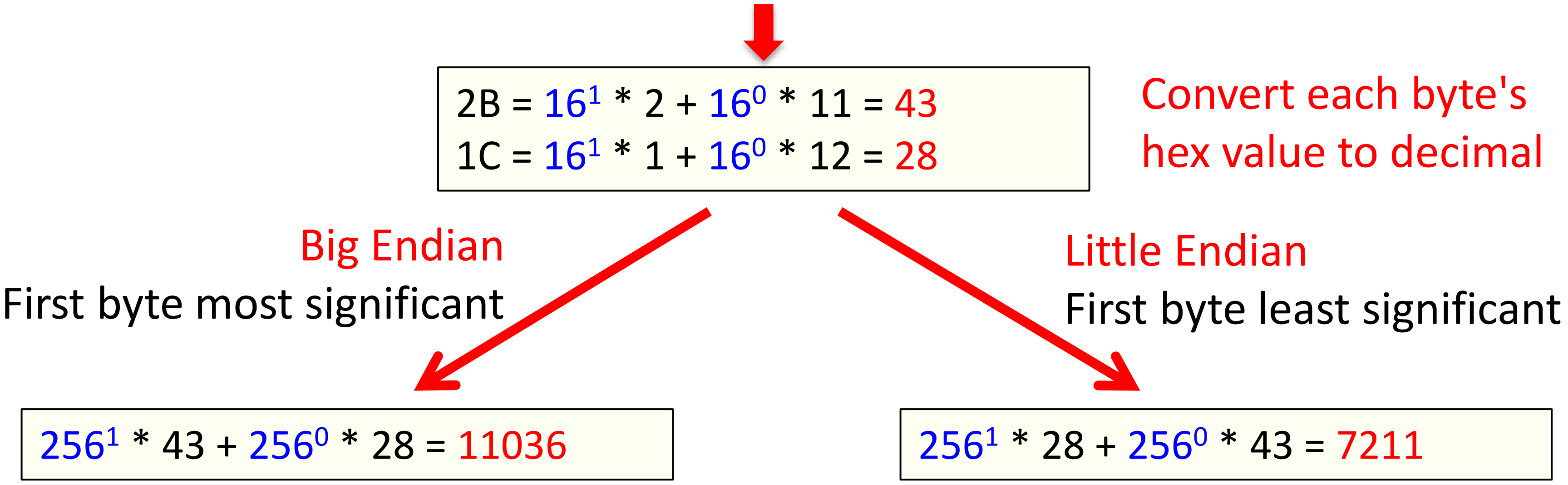
1C = $16^1 * 1 + 16^0 * 12 = 28$

Convert each byte's hex value to decimal

Conversion to decimal

- Goal: convert 2 bytes (16-bits) into decimal

Memory address	Hex value
1000h	2B
1001h	1C



2B = $16^1 * 2 + 16^0 * 11 = 43$

1C = $16^1 * 1 + 16^0 * 12 = 28$

Convert each byte's hex value to decimal

Big Endian

First byte most significant

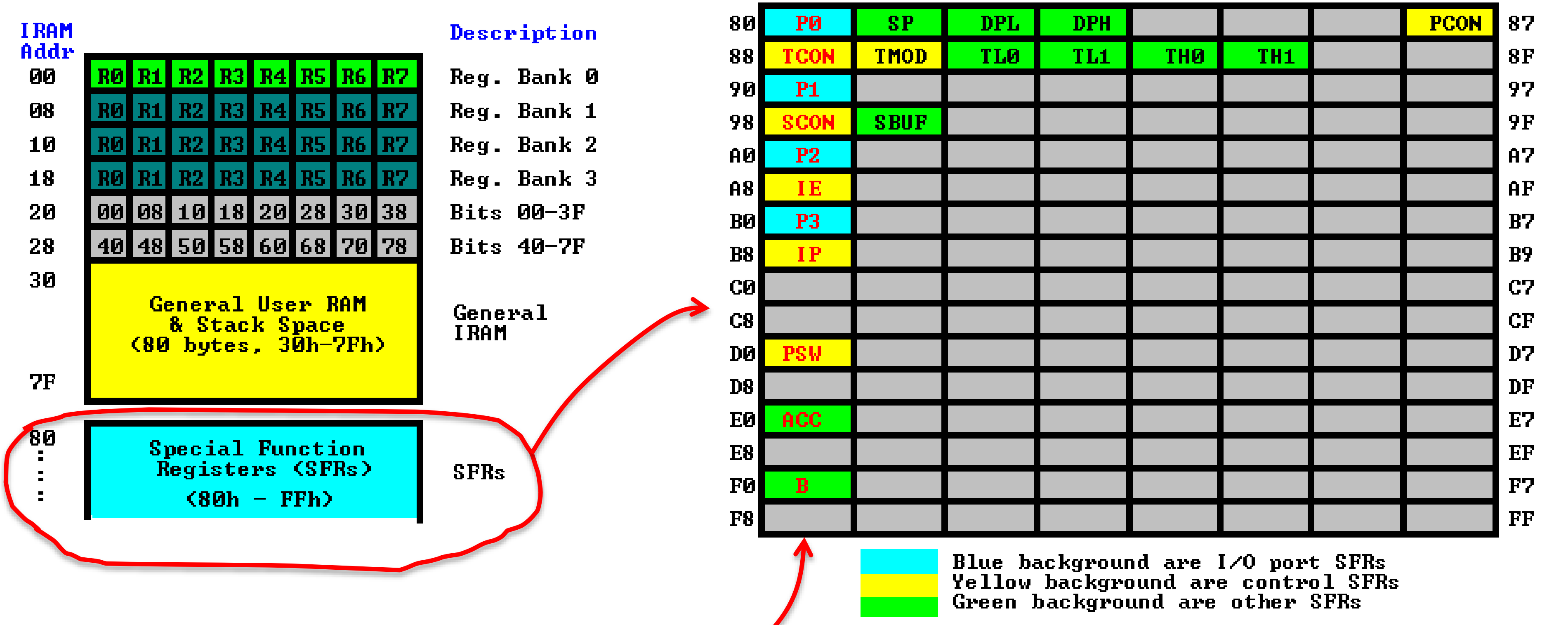
$256^1 * 43 + 256^0 * 28 = 11036$

Little Endian

First byte least significant

$256^1 * 28 + 256^0 * 43 = 7211$

Special Function Registers (SFRs)



IRAM Addr									Description
00	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 0
08	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 1
10	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 2
18	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 3
20	00	08	10	18	20	28	30	38	Bits 00-3F
28	40	48	50	58	60	68	70	78	Bits 40-7F
30	General User RAM & Stack Space (80 bytes, 30h-7Fh)								General IRAM
7F									
80 ⋮	Special Function Registers (SFRs) (80h - FFh)								SFRs

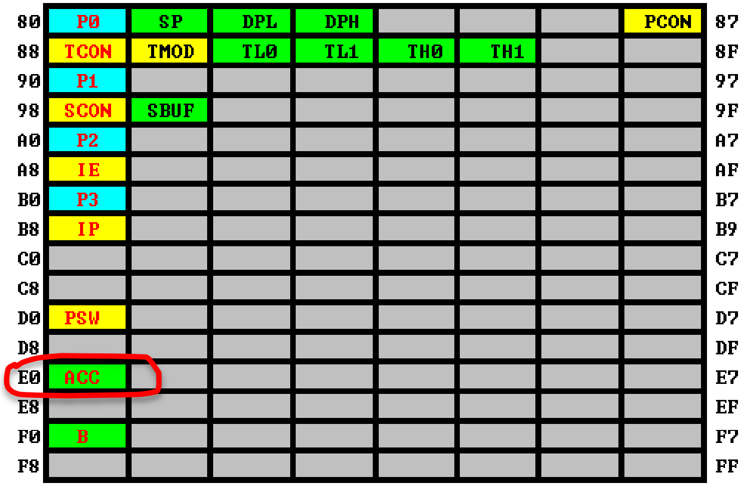
80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

🟦 Blue background are I/O port SFRs
🟨 Yellow background are control SFRs
🟩 Green background are other SFRs

SFRs with an address divisible by 8 are bit addressable (this column)

Accumulator (ACC)

- ACC
  - Address: E0h
  - Default value: 00h
  - Bit-addressable
  - Used in a majority of 8052 instructions
  - Usually referred to as "A", but "ACC" when modifying specific bits or using PUSH/POP

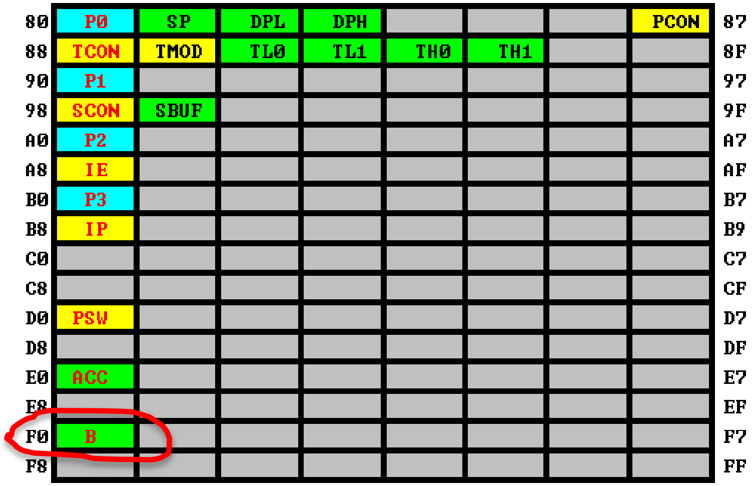


80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
Bit addr	E0h	E1h	E2h	E3h	E4h	E5h	E6h	E7h

B register (B)

- B
  - Address: F0h
  - Default value: 00h
  - Bit-addressable
  - Additional holding area, used in MUL/DIV

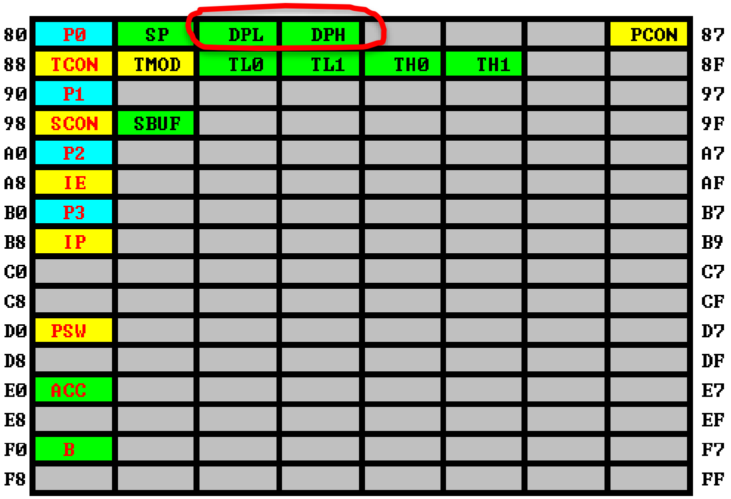


80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
Bit addr	F7h	F6h	F5h	F4h	F3h	F2h	F1h	F0h

Data pointer (DPTR)

- DPTR
  - Only 16-bit register
    - Composed of two SFRs
  - DPH
    - High byte
    - Address: 83h
  - DPL
    - Low byte
    - Address: 82h
  - Used to access code memory (MOVC) or external memory (MOVX)



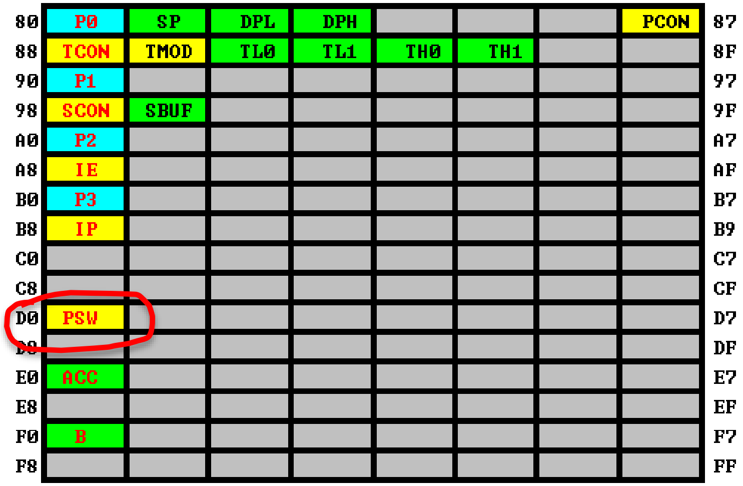
80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

Data pointer (DPTR)

- Can only be used in a few instructions:
  - INC DPTR
    - Increment the 16-bit value by one, 2 cycles
  - JMP @A+DPTR
    - Jump to address (implement a jump list)
    - Address is sum of DPTR and A
  - MOVC A,@A+DPTR
    - Move byte of code memory to accumulator
    - Address sum of DPTR and A
  - MOVX A,@DPTR and MOVX @DPTR, A
    - External memory access

Program Status Word (PSW)

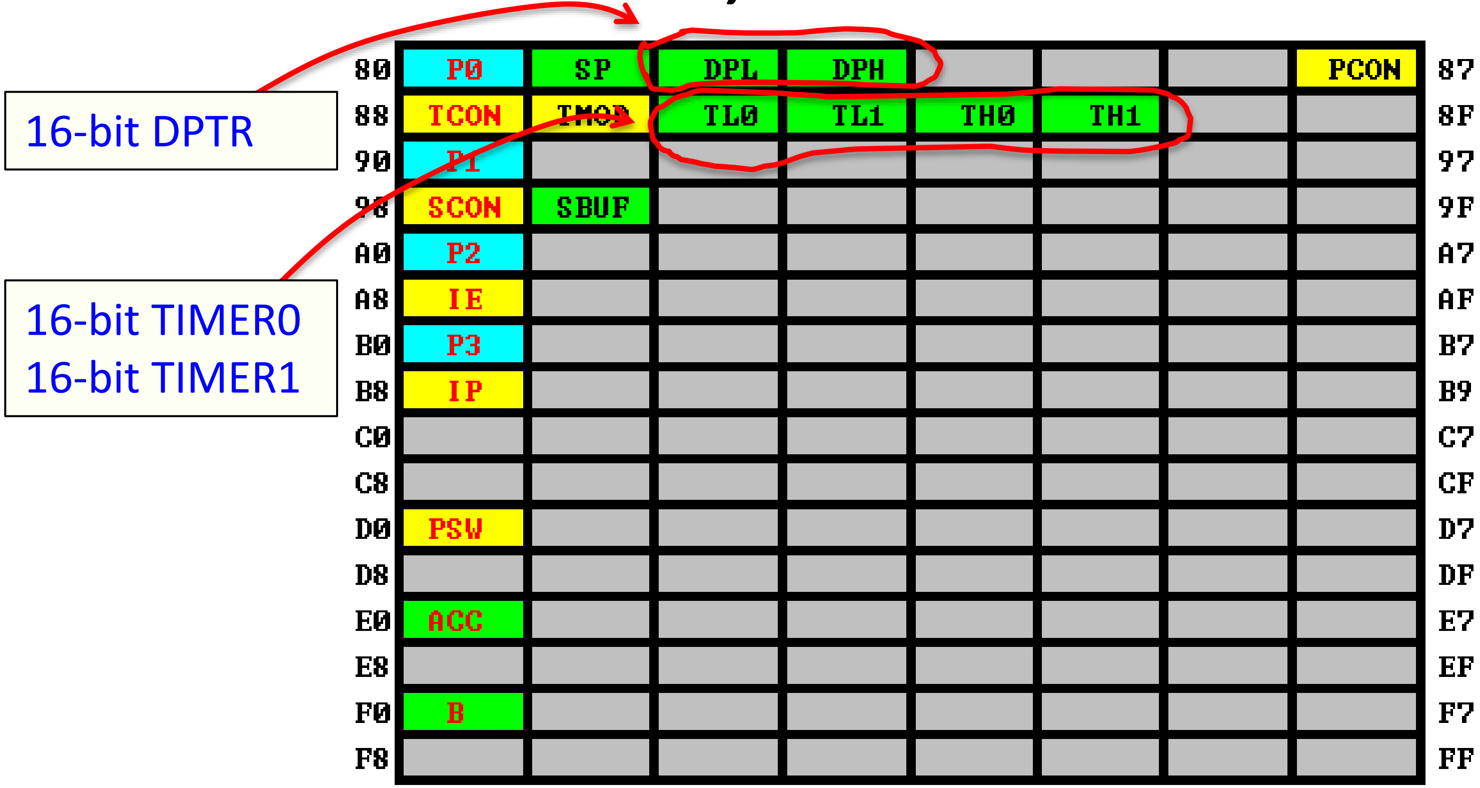
- PSW
  - Address: D0h
  - Bit addressable
  - Select register bank
  - Bits set and cleared by various math instructions
    - ADD, ADDC, SUBB, MUL, DIV



80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	CY	AC	F0	RS1	RS0	OV	-	P
Bit addr	F7h	F6h	F5h	F4h	F3h	F2h	F1h	F0h
	Carry flag	Aux. carry	General flag	Bank select	Bank select	Overflow flag		Even ACC parity

8052, Endianness?



16-bit DPTR

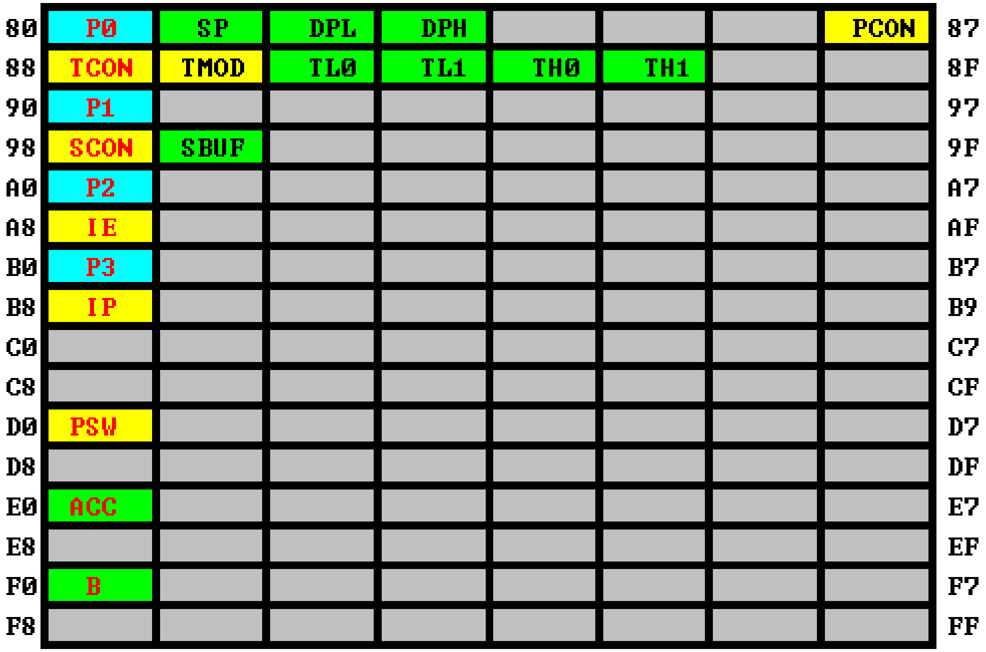
16-bit TIMER0
16-bit TIMER1

80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
Bit addr	E0h	E1h	E2h	E3h	E4h	E5h	E6h	E7h

Other SFRs

- Timer related:
  - Timer 0 (TH0/TL0), Timer 1 (TLH1/LH1), Timer 2 (TH2/TL2)
  - Timer Control (TCON), Timer Mode (TMOD)
  - Timer2 Control (T2CON), Reload/capture Timer 2 (RCAP2H/RCAP2L)
- Interrupt related:
  - Interrupt Enable (IE)
  - Interrupt Priority (IP)
- Serial communication:
  - Power Control (PCON)
  - Serial Buffer (SBUF)
  - Serial Control (SCON)



80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

Summary

- Byte order
  - How we split a big data type into chunks we can store in memory or send over a wire
- 8052 Special Function Registers (SFRs)
  - Stored at memory locations 80h and above
  - General purpose registers, ACC and B
  - Reading/writing from ports
  - Stay tuned:
    - Controlling timers, interrupts, serial communication