Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

Politechnika Śląska jako Centrum Nowoczesnego Kształcenia
opartego o badania i innowacje

POWR.03.05.00-IP.08-00-PZ1/17

Projekt współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego

Microprocessor and Embedded Systems

Faculty of Automatic Control, Electronics and Computer Science, Informatics, Bachelor Degree

Lecture 15

8051 single-chip microcomputer
Part 3
Programming basics

Bartłomiej Zieliński, PhD, DSc

8051 (3)

Program:

- Command groups
- Addressing modes
- Idata structure
- Command list
- Example assembler program
- Example C program

8051 (3)

- Basic properties
  - 111 commands (49 1B, 45 2B, 17 3B)
    - Transfer: idata↔idata (RAM, SFR), const→idata, A↔xdata, A←prog
    - Arithmetical/logical: add, sub, mul, div, shift, logical (or, and, xor operate on idata directly);
      - unsigned numbers, simple calc on signed (U2) and BCD
    - Jumps
      - Conditional – depending on A, CY, bit (idata), cmp (A, Ri, idata)
      - Unconditional
      - To subroutines (procedure calls)
    - Bit: and, or, not bit (idata)
      - Bit accumulator = CY
    - No I/O or control commands → SFR commands

8051 (3)

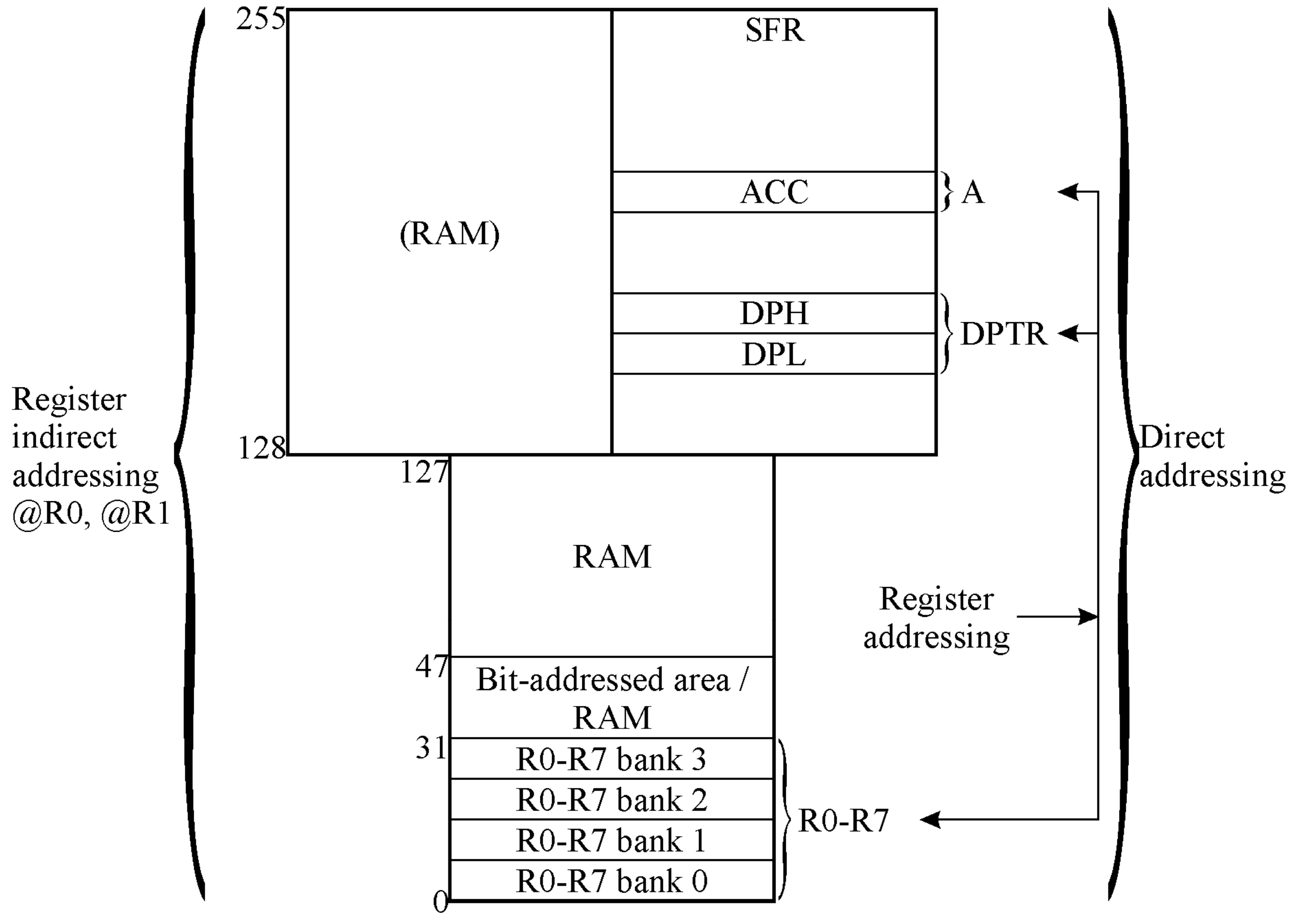
- Addressing modes
  - PROG
    - Immediate (direct argument)
      - E.g., lcall, ljmp
    - Indirect (16-b base reg + A)
      - Movc only
  - Jumps
    - Direct
      - 16-b address (within 64K: ljmp, lcall)
      - 11-b address (within 2K page: ajmp, acall)
    - Relative to PC
      - 8-b offset in U2 (-128..+127 bytes from next command)
    - Indirect
      - (DPTR + A)

8051 (3)

- Addressing modes
  - IDATA
    - By register name (R0..R7, ACC, DPTR)
    - Direct (8-bit addr in command)
      - SFR 128...255, memory 0…127
    - Register indirect
      - @R0, @R1: 0…127(255)
    - Direct bit
      - IDATA addresses 32-47, some SFR
  - XDATA
    - Indirect (`movx` command)
      - @R0, @R1 – 8-bit address
      - @DPTR – 16-b address

8051 (3)

- Internal data memory organisation

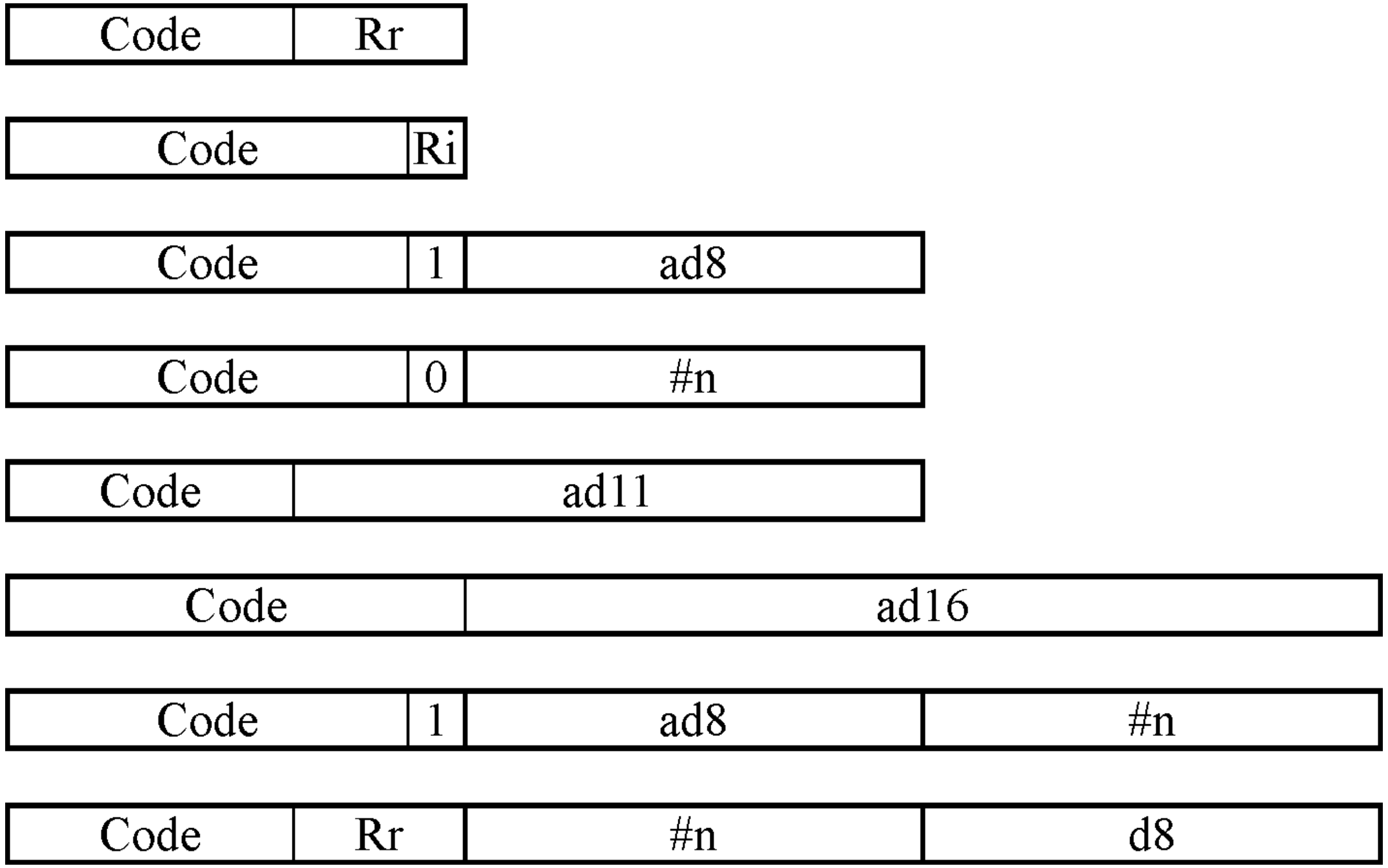


8051 (3)

- Command list
  - Notation:
    - Rr = R0…R7
    - Ri = R0…R1
    - Ad = 8-b direct address
    - n = 8-b direct argument
    - nn = 16-b direct argument

8051 (3)

- Command formats



8051 (3)

- Command list – data transfer
  - mov (8-b transfer)
    - A, Rr / Rr, A
    - A, ad / ad, A
    - A, @Ri / @Ri, A
    - A, #n
    - Rr, ad / ad, Rr
    - Rr, #n
    - ad1, ad2
    - ad, @Ri / @Ri, ad
    - @Ri, #n
    - ad, #n
  - mov (16-b transfer)
    - dptr, #nn

8051 (3)

- Command list – data transfer
  - xch
    - A, Rr
    - A, ad
    - A, @Ri
    - Xchd A, @Ri; $A_{0..3} \leftrightarrow (Ri)_{0..3}$
  - movx
    - A, @Ri / @Ri, A
    - A, @dptr / @dptr, A
  - movc
    - A, @A+DPTR; A=(A+DPTR)
    - A, @A+PC; A=(A+PC)

8051 (3)

- Command list – arithmetical/logical
  - Add/addc/subb
    - A, Rr
    - A, ad
    - A, @Ri
    - A, #n
  - Anl/orl/xrl
    - A, Rr
    - A, ad
    - A, @Ri
    - A, #n
    - Ad, A
    - Ad, #n

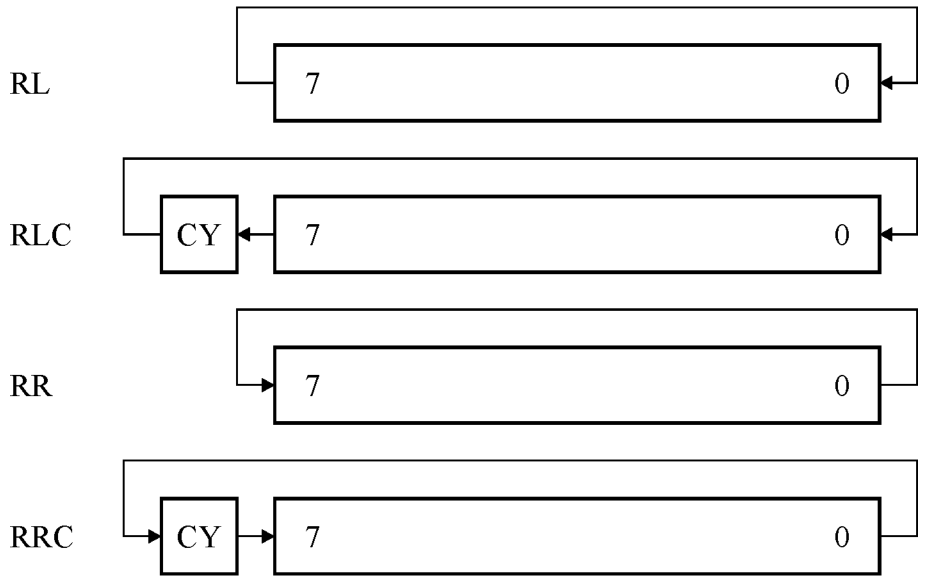
8051 (3)

- Command list – arithmetical
  - inc/dec
    - A
    - Rr
    - Ad
    - @Ri
  - Inc DPTR
  - Mul ab; b.a=a×b
  - Div ab; a=a/b (no remainder)
  - Da a; decimal correction

8051 (3)

- Command list
  - Clr, swap, cpl, rl, rlc, rr, rrc
    - A

    - Clr a; A=0
    - Swap; $A_{0..3} \leftrightarrow A_{4..7}$
    - Cpl; complement



8051 (3)

- Command list – bit manipulation
  - Clr/setb/cpl
    - C
    - Bit
  - And/orl
    - C, bit
    - C, /bit
  - Mov
    - C, bit
    - Bit, C

8051 (3)

- Command list – jumps
  - Ajmp adr11
  - Ljmp adr16
  - Sjmp d8
  - Jmp @A+DPTR
  - Jc/jnc/jz/jnz d8
  - Jb/jnb/jbc   bit, d8
  - Cjne
    - A, ad, d8
    - A, #n, d8
    - Rr, #n, d8
    - @Ri, #n, d8
  - Djnz
    - Rr, d8
    - Ad, d8

8051 (3)

- Command list
  - Calls/returns/stack
    - Acall adr11
    - Lcall adr16
    - Ret
    - Reti
    - Push /pop   ad
  - And finally the most important command in every microprocessor
    - NOP

8051 (3)

- Example program
  - „watchdog” expansion card for ISA-bus equipped microcomputer (e.g., IBM PC until abt. 2000)
  - 8051 as a peripherial µp
  - Communication using 2 registers
    - PC→8051 (command)
    - 8051→PC (status)
    - Both available for 8051 at any address
    - Interrupt-driven communication
  - Fully software-defined „commands”

8051 (3)

```
; ***************************************************************************
;                    Registers usage:
;                              - R0: minutes of current Timeout constant
;                              - R1: minutes of Timeout 1
;                              - R2: minutes of Timeout 2
;                              - R3: seconds of current Timeout
;                              - R4: sec/100 of current Timeout
; ***************************************************************************

TimerHi    EQU 0D8h
TimerLo    EQU 0EFh

ORG 0000h                                  ; power-on reset
           ljmp Init

ORG 0003h                                  ; external interrupt
           lcall Command
           reti

ORG 000Bh                                  ; timer 0 overflow interrupt
           lcall Counter
           reti

ORG 0040h
```

8051 (3)

```
; *************************************************************************
;                         Microcontroller initialization:
;                                     - clear interrupt flip-flop
;                                     - enable ext. int. 0
;                                     - set OK status
; *************************************************************************

Init:
        clr  p1.0                       ; out of Reset
        movx a, @r0                     ; clear ext. interrupt flip-flop
        mov  tmod, #00000001b           ; counter 0: mode 1, timer, prog. cntl
        mov  ie,  #10000011b            ; enable all ints, ext 0, timer 0
        clr  a
        movx @r0, a                     ; set NOP status

Itself:
        ljmp Itself                     ; remain here forever
```

8051 (3)

```
; ***************************************************************************
;                                   Timer 0 interrupt service:
;                                                   - if end of t1, set timeout 1 status
;                                                   - if end of t2, set timeout 2 status & send reset
; ***************************************************************************
Counter:
                mov  th0, #TimerHi
                mov  tl0, #TimerLo
                djnz r4, Return                                 ; sec/100
                mov  r4, #100
                djnz r3, Return                                 ; seconds
                mov  r3, #60
                djnz r0, Return                                 ; minutes
                jb              f0, Timeout2                    ; f0 = 1 => Timeout2

Timeout1:
                mov  a, #02h
                movx @r0, a                                     ; set "Time 2 running" status
                setb f0                                         ; sign Timeout1
                mov  a, r2
                mov  r0, a                                      ; copy Timeout2 constant
Return:
                ret
```

8051 (3)

```
; ******************************************************************************
;                         Timer 0 interrupt service:
;                                     - if end of t1, set timeout 1 status
;                                     - if end of t2, set timeout 2 status & send reset
; ******************************************************************************
Timeout2:
            mov  a, #66h
            movx @r0, a                 ; set "Master Reset" status
            setb p1.0                   ; send reset pulse

WaitReset:
            jnb  p1.1, WaitReset        ; wait until Reset active

            clr  p1.0                   ; clear reset
            clr  tr0                    ; stop timer 0
            clr  f0                     ; sign Timeout1
            ret
```

8051 (3)

```
; ***************************************************************************
;  External Interrupt 0 service:
;               - receive & execute command:
;                           - x1h: Time1: set Timeout 1 period as x
;                           - x2h: Time2: set Timeout 2 period as x
;                           - 03h: StartCnt: start counter with Timeout 1, enable counter 0 int.
;                           - 04h: StopCnt: stop counter, disable counter 0 int.
;                           - 05h: ResetCnt: start counter with Timeout 1
;                           - anything else is bad command, so we ignore it
; ***************************************************************************
Command:
                movx a, @r0                         ; receive command
                mov  b, a                           ; store byte for future use
                anl  a, #0Fh                        ; command code

ChkTime1:
                cjne a, #1, ChkTime2
                call CalcTime
                mov  r1, a
                ret

ChkTime2:
                cjne a, #2, ChkStart
                call CalcTime
                mov  r2, a
                ret
```

8051 (3)

```
ChkStart:
            cjne a, #3, ChkStop
            mov  th0, #TimerHi
            mov  tl0, #TimerLo
            mov  r4, #100
            mov  r3, #60
            mov  a, r1
            mov  r0, a              ; choose Timeout1
            clr  f0                 ; sign Timeout 1
            mov  a, #01h
            movx @r0, a                         ; set "Time 1 running" status
            setb tr0                ; enable timer 0
            ret
ChkStop:
            cjne a, #4, ChkReset
            clr  tr0                ; disable timer 0
            clr  a
            movx @r0, a                         ; set "NOP" status
            ret
```

8051 (3)

```
ChkReset:
        cjne a, #5, IgnCommand
        mov  th0, #TimerHi
        mov  tl0, #TimerLo
        mov  r4, #100
        mov  r3, #60
        mov  a, #01h
        movx @r0, a                     ; set "Time 1 running" status
        mov  a, r1
        mov  r0, a                      ; choose Timeout 1
        clr  f0                         ; sign Timeout 1
        ret
IgnCommand:
        ret
```

8051 (3)

```
#pragma language=extended
#include <io51.h>

/*******************************************************************/

#define TIMER_HI 0xD8
#define TIMER_LO 0xEF

/*******************************************************************/

idata char time1, time2;
bit timeout2;
xdata char status;
xdata volatile char command;
idata char hun, sec, min;

/*******************************************************************/
```

8051 (3)

```
interrupt [0x03] void ext_int (void) {
  register char cmd, par;

  cmd = command;
  par = (cmd >> 4) & 0x0F;
  cmd &= 0x0F;

  switch (cmd) {
    case 1:
      time1 = par;
      break;
    case 2:
      time2 = par;
      break;
    case 3:
      TH0 = TIMER_HI;
      TL0 = TIMER_LO;
      sec = ((min = time1) != 0) ? 60 : 5;
      hun = 100;
      timeout2 = 0;
      status = 1;
      TR0 = 1;
      break;
    case 4:
      TR0 = 0;
      status = 0;
      break;
    case 5:
      TH0 = TIMER_HI;
      TL0 = TIMER_LO;
      sec = ((min = time1) != 0) ? 60 : 5;
      hun = 100;
      status = 1;
      timeout2 = 0;
      break;
    default:
      break;
  } /* switch */

} /* ext_int */
```

8051 (3)

```
interrupt [0x0B] void clk_int (void) {
  TH0 = TIMER_HI;
  TL0 = TIMER_LO;

  if (--hun)
    return;
  hun = 100;
  if (--sec)
    return;
  sec = 60;
  if ((time1 || time2) && --min)
    return;

  if (!timeout2){
    status = 2;
    timeout2 = 1;
    sec = ((min = time2) != 0) ? 60 : 5;
  } else {
    status = 0x66;
    P1.0 = 1;
    while (!P1.1);
    P1.0 = 0;
    TR0 = 0;
    timeout2 = 0;
  } /* if */

  return;
} /* clk_int */

/*********************************/

void main (void) {

  P1.0 = 0;
  ACC = command;
  TMOD = 0x01;
  IE = 0x83;
  status = 0;
  while (1);

} /* main */
```