Wykład 11

Mikrokomputery jednoukładowe rodziny PIC
Część 1
Architektura

Bartłomiej Zieliński, PhD, DSc

PIC (1)

Program:

(dziś)

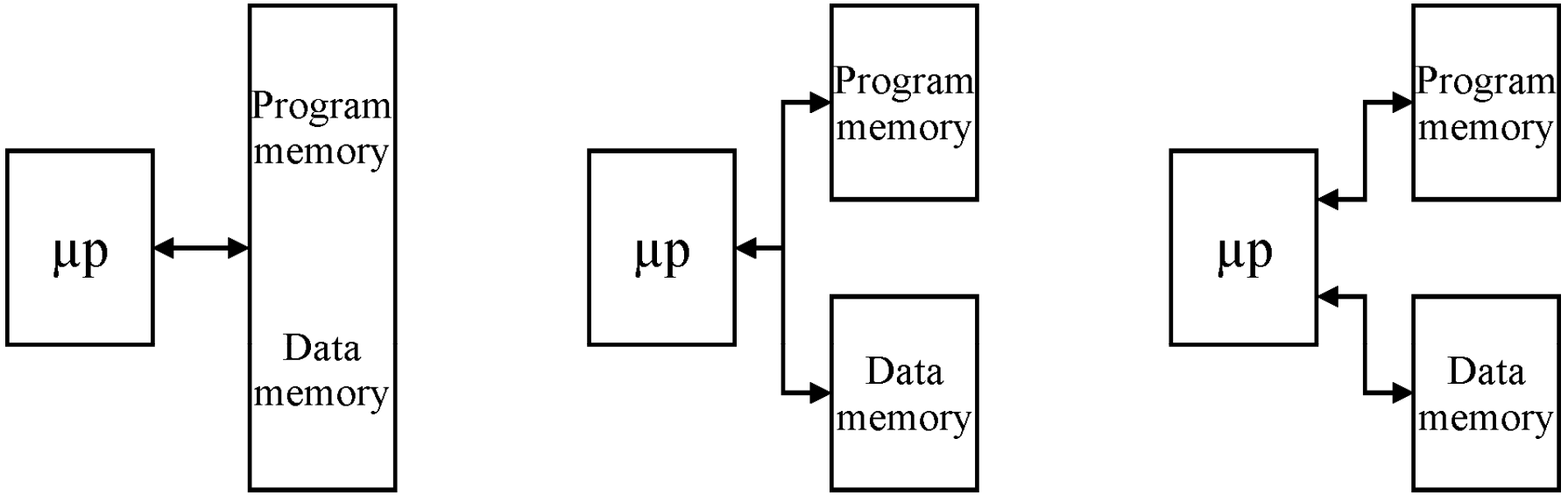
- Architektura von Neumanna i harwardzka
- Struktura, wyprowadzenia PIC
- Organizacja pamięci, adresowanie

(za tydzień)

- Wbudowane układy we-wy

PIC (1)

- Architektura von Neumanna i harwardzka



von Neumann architecture

"Intermediate" architecture

Harvard architecture

PIC (1)

- Architektura von Neumanna i harwardzka
  - Oddzielne magistrale programu i danych
    - Długość słowa pamięci programu ≠ długość słowa pamięci danych
      - Kod rozkazu = 1 słowo pamięci programu
      - Cały kod i argumenty rozkazu w pojedynczym słowie
      - Długość i format kodu może być różna w różnych wersjach
        - » Zgodność kodu źródłowego
    - Niezależne pamięci programu i danych
      - Jednoczesny dostęp
        - » Np. rozkaz 1 zapisuje, gdy rozkaz 2 jest pobierany
        - » Szybsze wykonanie programu → potokowość
    - Niejednoznaczność adresów
      - Trzeba określić przestrzeń adresową programu/danych

PIC (1)

- Rodziny PIC

Rodzina	Program	Rozkazy	Przykłady
Base-line	12 b	33	PIC12Cxxx, PIC16Cx
Mid-Range	14 b	35	PIC16C/Fxxx, PIC12C6xx
High End	16 b	55	PIC17xxx
Enhanced	16 b	77	PIC18xxx
dsPIC			

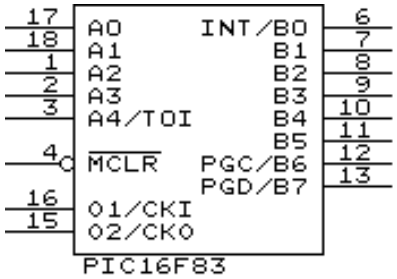
- Wersje
  - F = Flash
  - C = OTP EPROM
  - CR = ROM
  - CE = OTP + EEPROM
  - HV = High voltage (15V)
  - L = Low voltage (2÷5.5V)
  - E.g., LF, LCR, etc.

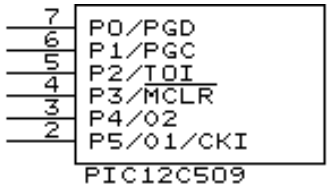
PIC (1)

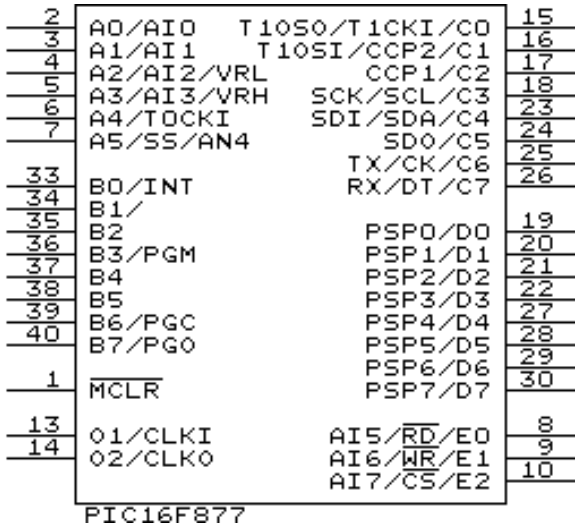
- Rodziny PIC
  - Obudowy 8, 18, 20, 28, 40, 64, 84 wyprowadzeń
  - Wbudowane moduły we-wy:
    - EEPROM
    - ADC, DAC
    - Komparatory analogowe
    - USART, SPI, I^2C, CCP
    - LIN, CAN, USB
    - Czujniki temperatury
    - Wzmacniacz operacyjny
    - …

PIC (1)

- PIC – przykładowe wyprowadzenia

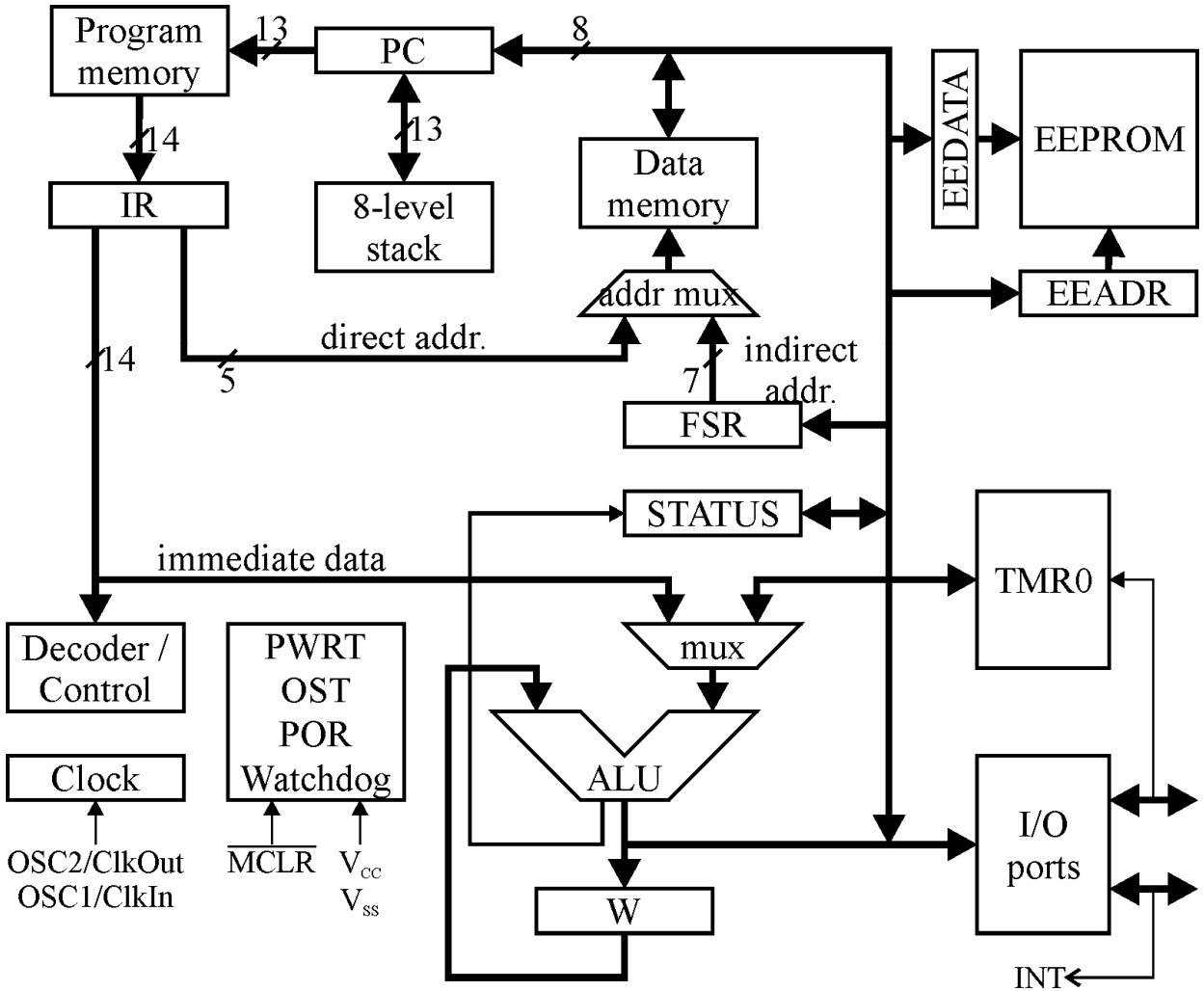






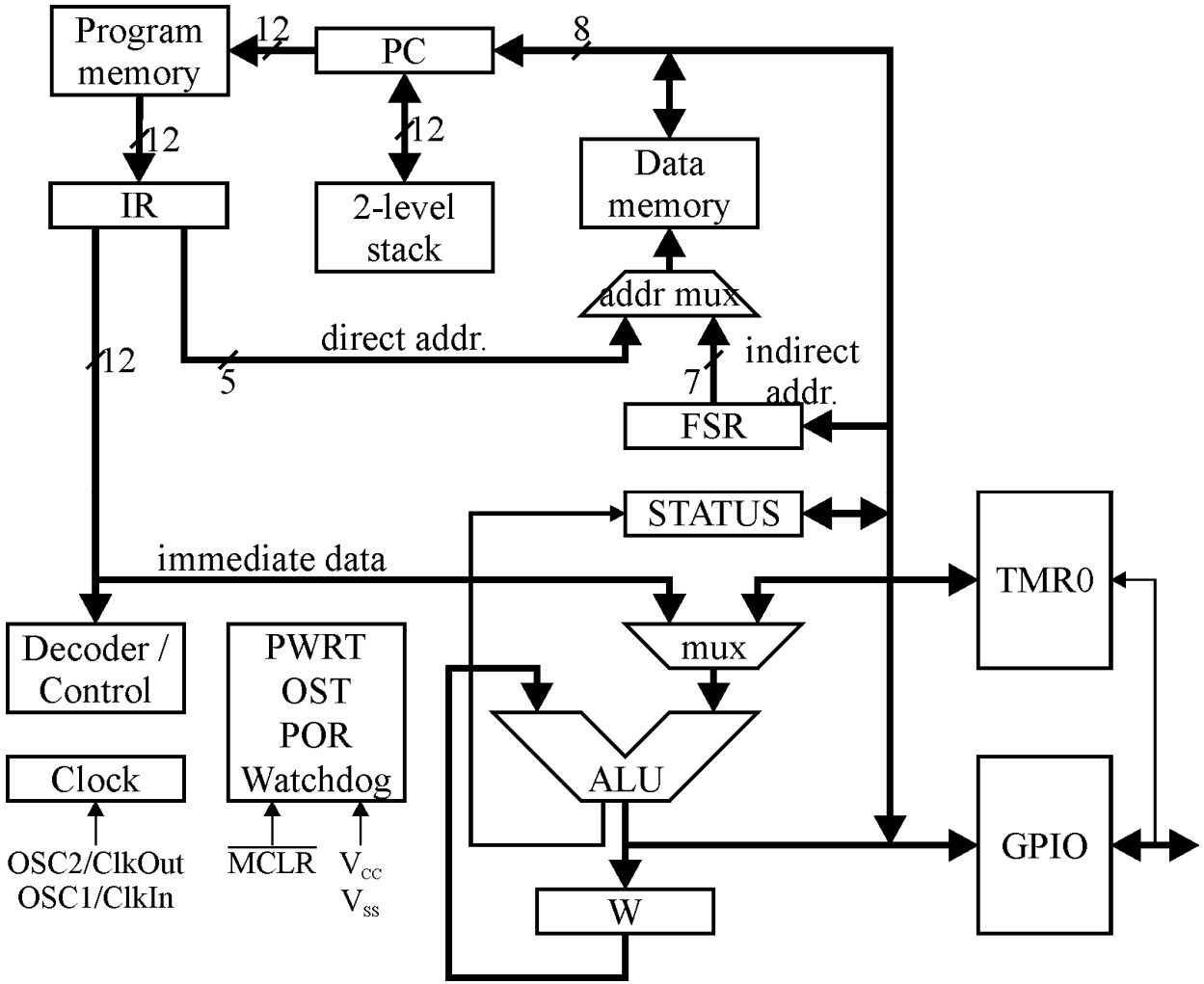
PIC (1)

- PIC16F8x – ogólna architektura



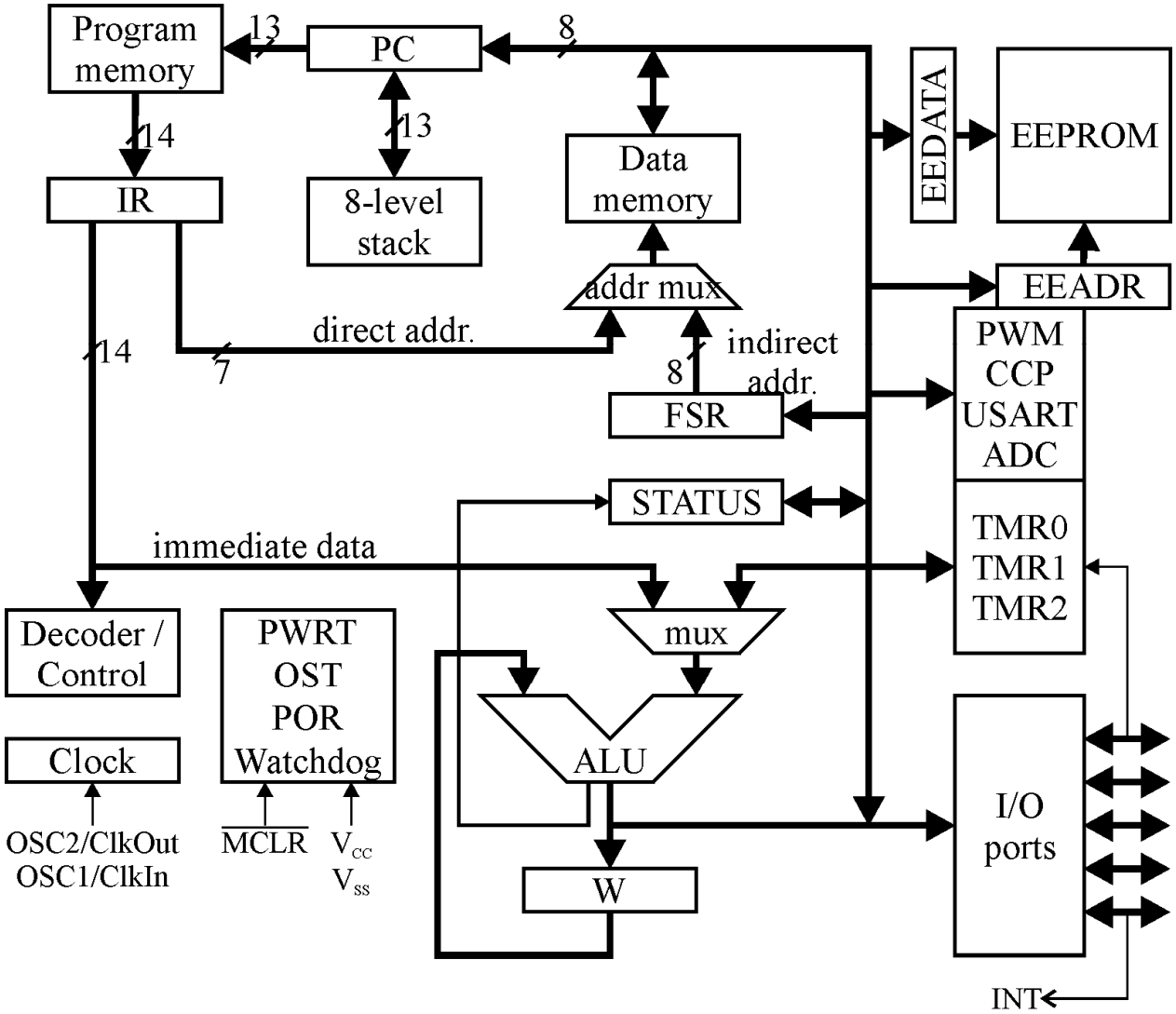
PIC (1)

- PIC12C509 – ogólna architektura



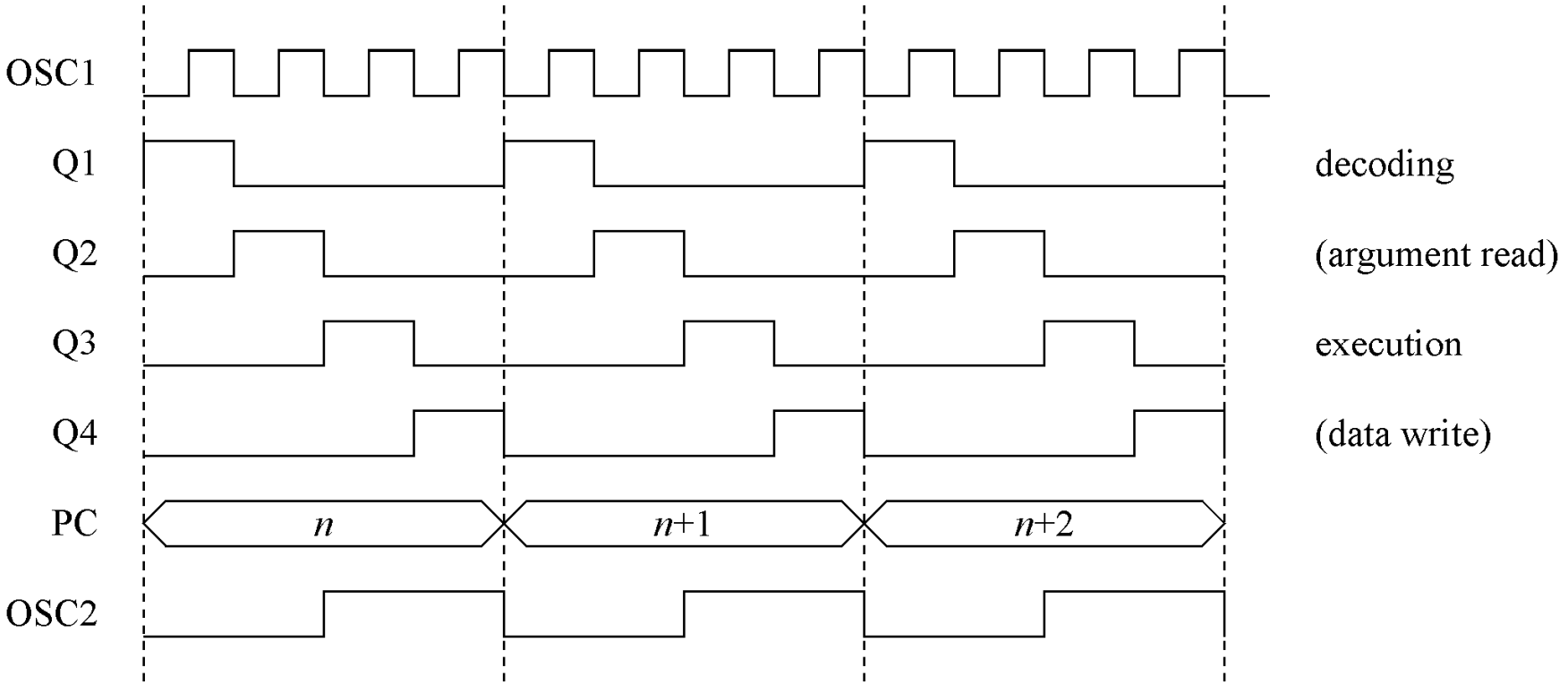
PIC (1)

- PIC16C877 – ogólna architektura



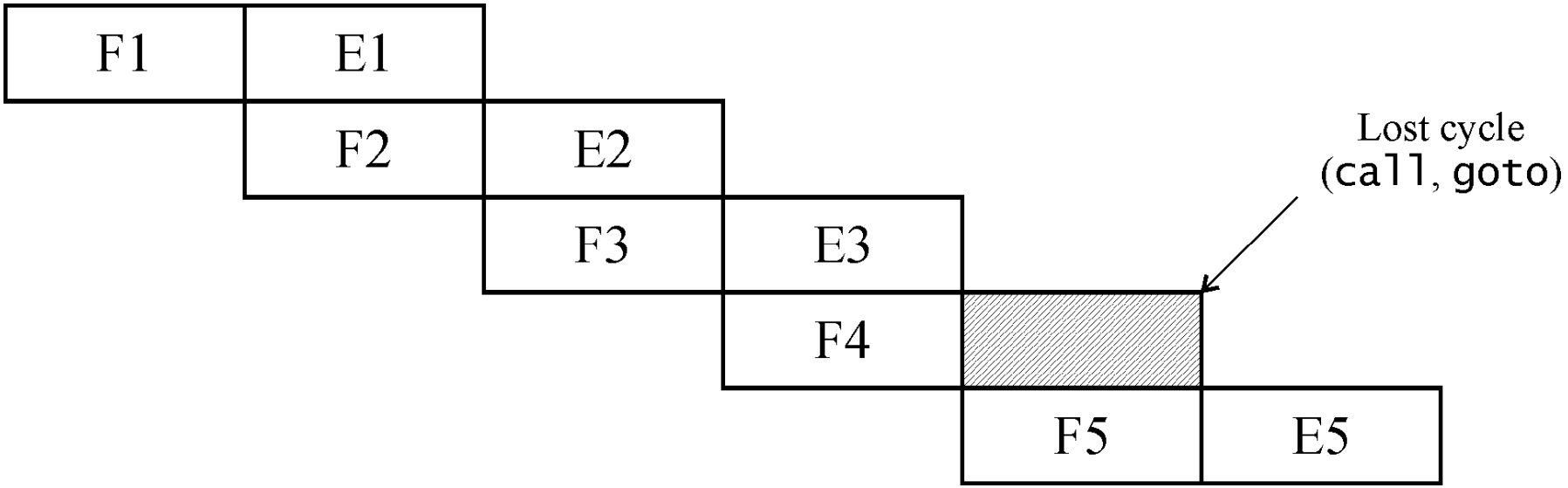
PIC (1)

- PIC16F8x – cykl rozkazowy
  - 4 cykle maszynowe Q1÷Q4
  - 1 cykl maszynowy = 1 cykl zegarowy
    - Np. 4 MHz → 1µs/rozkaz



PIC (1)

- PIC16F8x – cykl rozkazowy



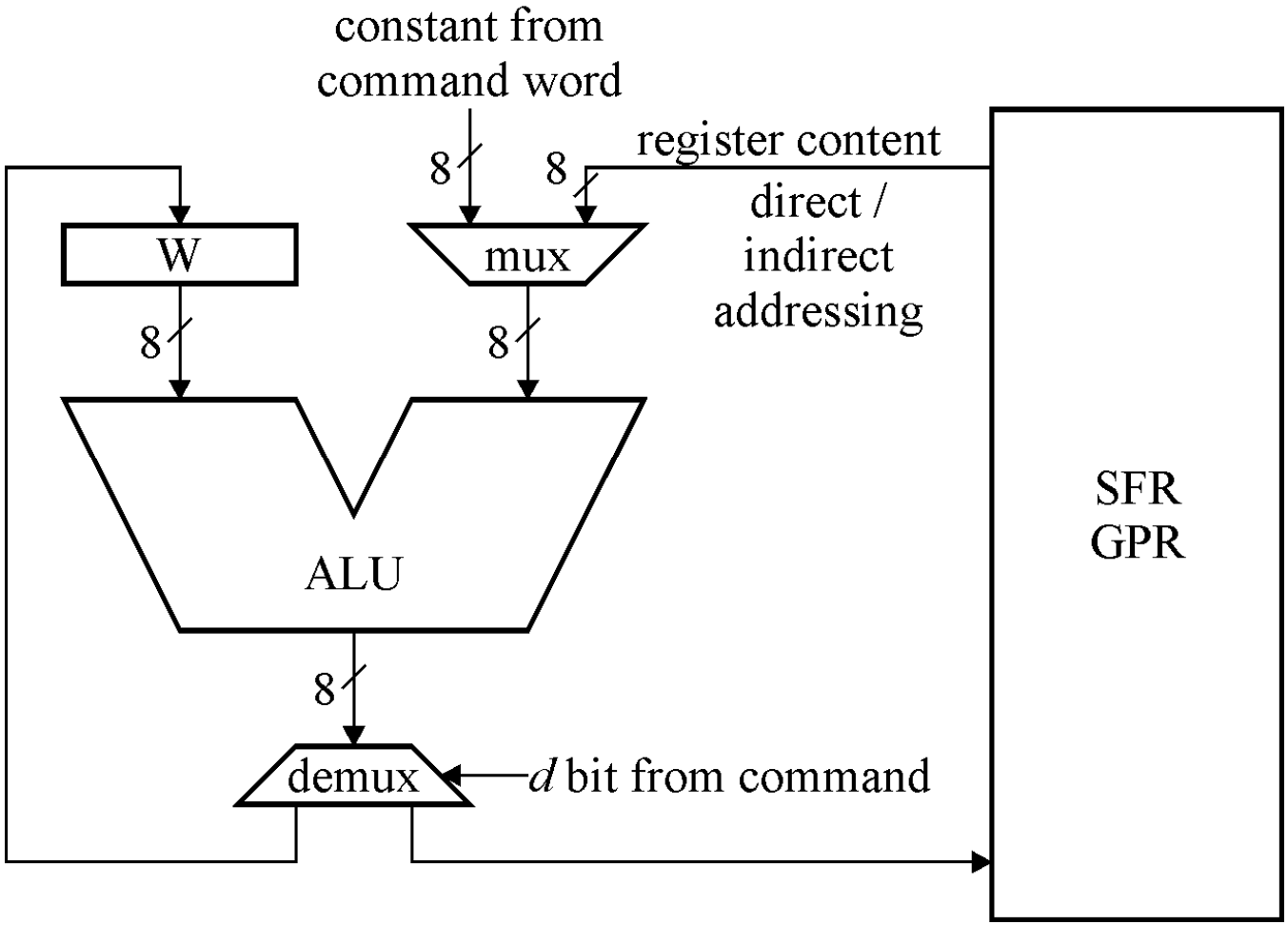
  - 1 rozkaz = 2 cykle rozkazowe
    - Prócz `call`, `goto` i innych działających na PC

PIC (1)

- PIC16F8x – operacje JAL
  - Rejestry
    - Rejestr W (*akumulator czy nie?*)
    - GPR/SFR
  - Argumenty i wynik operacji
    - Arg1 ← W
    - Arg2 ← SFR, GPR lub stała
    - Wynik → W, SFR, GPR
  - Modyfikacja znaczników (rejestr STATUS)
    - C – przeniesienie/pożyczka
    - DC – pomocnicze przeniesienie/pożyczka
    - Z – zero

PIC (1)

- PIC16F8x – operacje JAL



PIC (1)

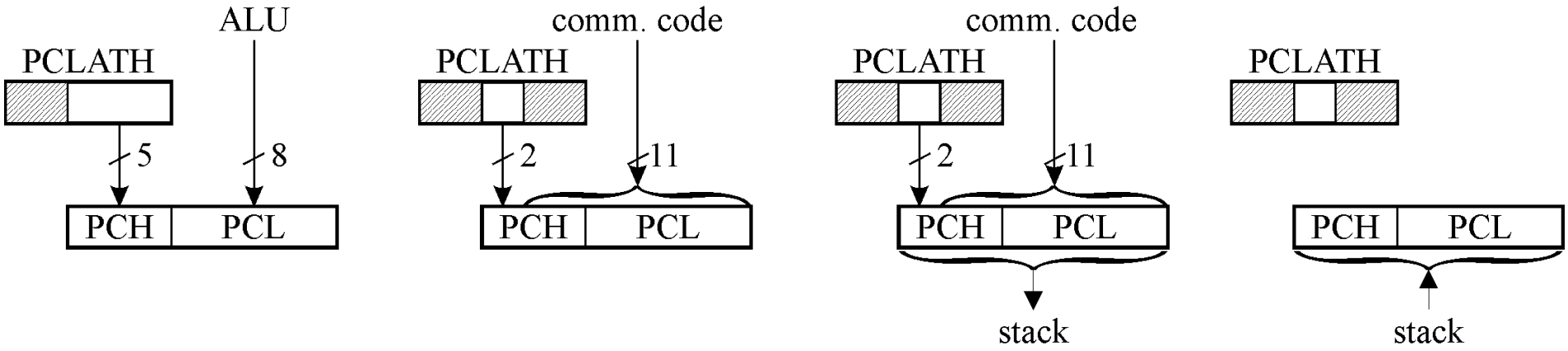
- PIC16F8x – operacje JAL
  - Rejestr STATUS
    - Można wpisać dowolną wartość
    - C, DC, Z ustawiane automatycznie
    - !TO, !PD – niemodyfikowalne
      - !TO=0 → przepełnienie układu nadzorującego (*watchdog*)
      - !PD=0 → po rozkazie `sleep`
    - IRP – wybór banku pamięci RAM (adr. pośrednie)
      - Nieużywane, gdy tylko 2 banki pamięci
    - RP0, RP1 – wybór banku pamięci RAM (adr. bezpośr.)
      - RP1 nieużywane, gdy mniej niż 3 banki pamięci

PIC (1)

- PIC16F8x – OPTION_REG
  - !RBPU
    - 0 → oporniki podciągające portu B włączone
  - INTEDG
    - Zgłaszanie przerwania zboczem narastającym (1)/ opadającym (0)
  - T0CS
    - Źródło taktowania T0: zewnętrzne (1)/wewnętrzne (0)
  - T0SE
    - Aktywne zbocze narastające (1)/opadające (0)
  - PSA
    - Preskaler przypisany do ukł. nadzorującego (1) /TMR0 (0)
  - PS2..0
    - Preskaler 1/2 … 1/256 (TMR0), 1/1 … 1/128 (u. nadzor.)

PIC (1)

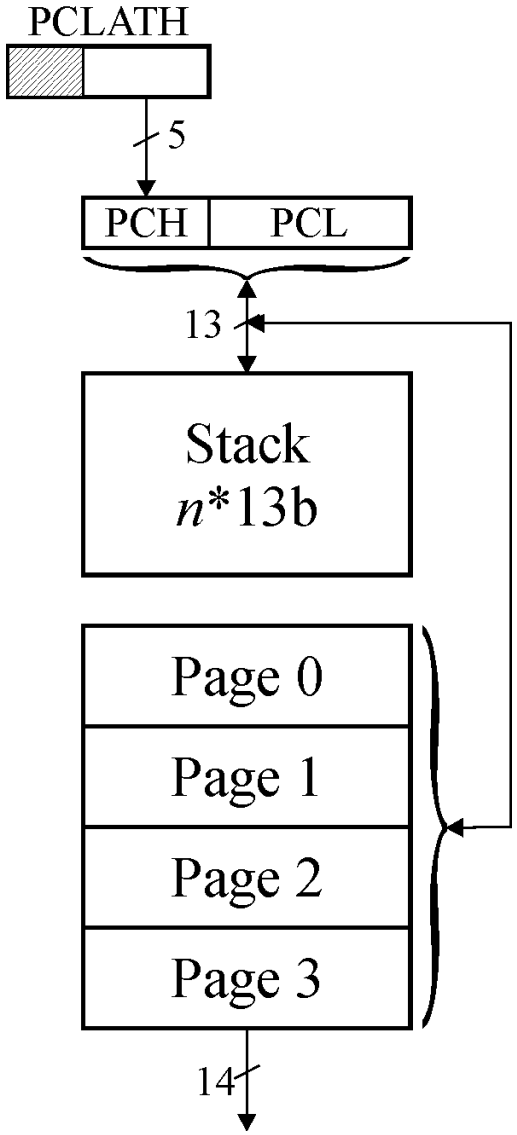
- PIC16F8x – licznik programu (PC)
  - $PC_{0..12}$:
    - PCL = $PC_{0..7}$
      - Można odczytywać i zapisywać
    - PCH = $PC_{8..12}$
      - Bezpośrednio niemodyfikowalny
      - Modyfikacja przez PCLATH



computed goto *goto* *call* *return*

PIC (1)

- PIC16F8x – pamięć programu
  - Wektor zerowania: 0000h
  - Wektor przerwania: 0004h
  - Stos: wyłącznie sprzętowy
    - Brak dostępu programowego
    - Brak rozkazów `push`, `pop`
    - Brak przekazywania argumentów
    - **Brak zgłaszania przepełnienia!**
  - Maks. pojemność = 8k×14
    - Może być mniejsza, niż 8k

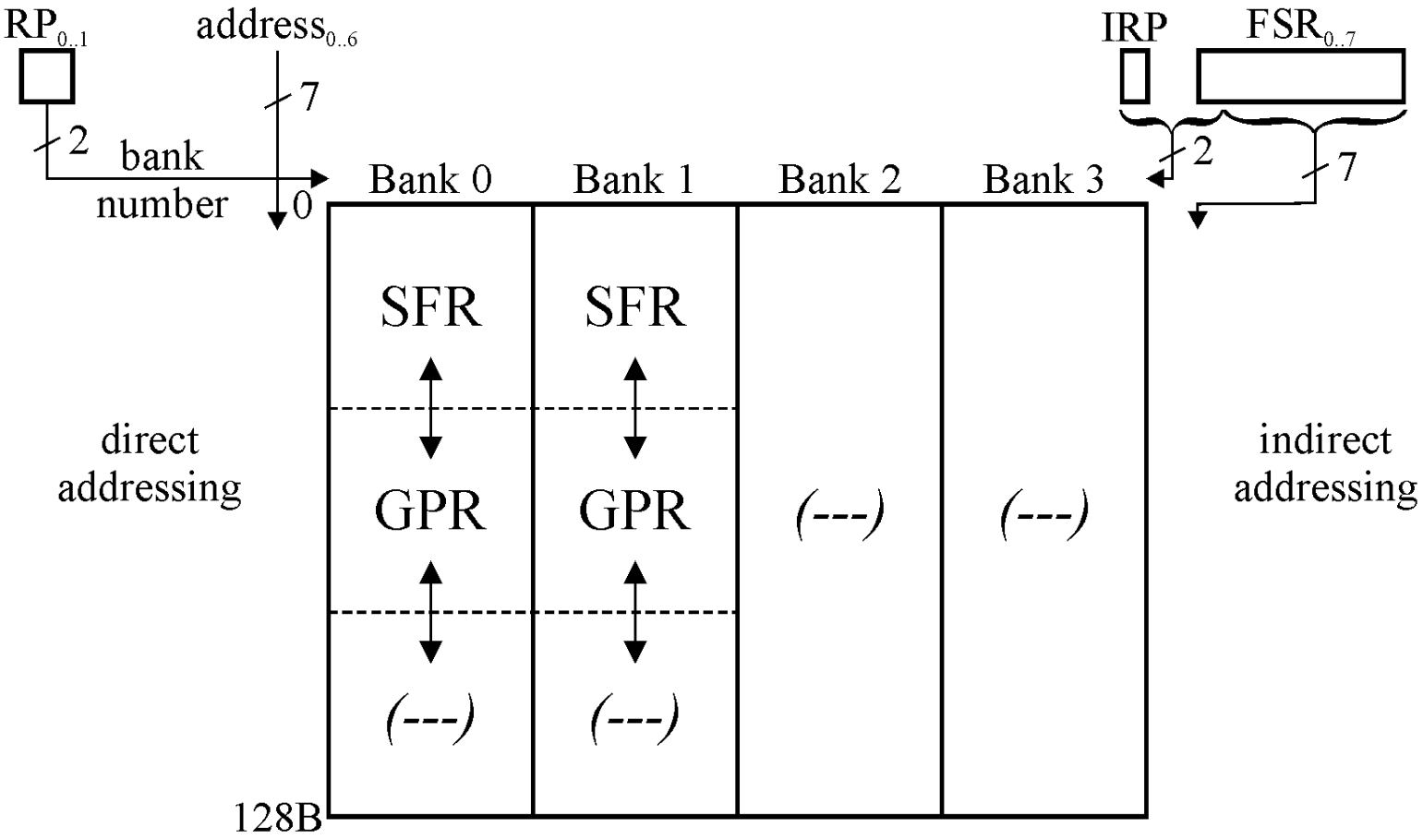


PIC (1)

- PIC16F8x – pamięć konfiguracyjna
  - Powyżej pamięci programu
  - Dostępna tylko podczas programowania
  - Opcje:
    - Rodzaj oscylatora
    - Wł/wył układu nadzorującego
    - Opóźnienie przy włączaniu zasilania
    - Blokada odczytu pamięci programu/EEPROM
      - Nie można odczytać z zewnątrz mikroprocesora

PIC (1)

- PIC16F8x – pamięć danych

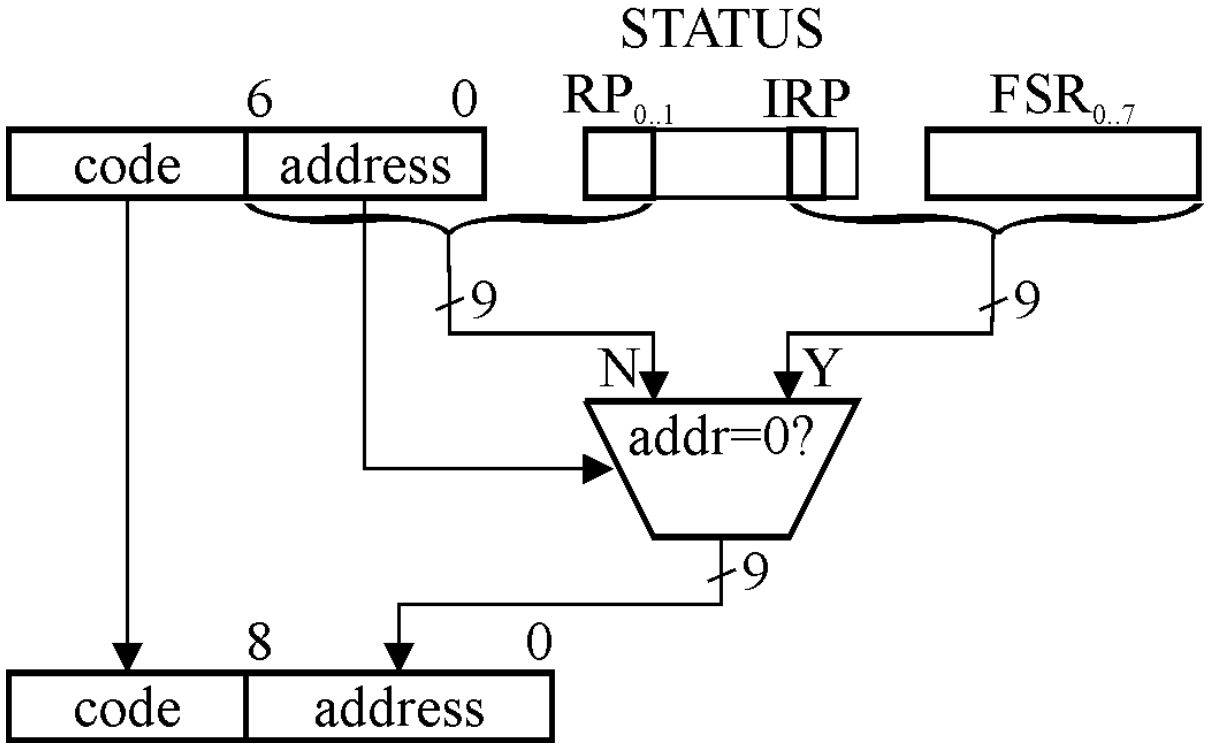


PIC (1)

- PIC16F8x – pamięć danych
  - Zależne od wersji:
    - Pojemność, liczba banków, granice obszarów
  - SFR:
    - Rejestry kluczowe
      - Mapowane w każdym banku
      - Stały adres
    - Pozostałe rejestry
      - Mapowane tylko w pojedynczym banku
  - GPR (bank 1) i GPR (bank 0)
    - Mogą być to fizycznie te same komórki!

PIC (1)

- PIC16F8x – adresowanie pamięci danych



  - Bepośredni dostęp do INDF = pośredni dostęp do GPR

PIC (1)

- PIC16F8x – adresowanie pamięci danych
  - Adresowanie pośrednie
    - Pseudo-rejestr INDF
      - Mapowany pod adresem 0 w każdym banku
      - Nie jest prawdziwym rejestrem
    - Odczyt:
      - Adres → IRP.FSR
      - INDF → dane
    - Zapis:
      - Adres → IRP.FSR
      - Dane → INDF

PIC (1)

- PIC16F8x – pamięć EEPROM
  - Adresowana tylko pośrednio
  - ≥ 10^7 cykli kasowania i zapisu
  - Rejestry:
    - EEDATA – 8-bit dane, Rd/Wr
    - EEADR – 8-bit adres
    - EECON1 – sterowanie odczytem i zapisem
      - WR, RD, WREN, WRERR, EEIF, EEPGD
    - EECON2 – „hasło” podczas zapisu
      - „*abrakadabra*”, „*hokus-pokus*” itp.

PIC (1)

- PIC16F8x – pamięć EEPROM

Read	Write
adres → EEADR EEPGD = 0 EECON1.RD = 1 (Δt = 0) EEDATA → dane	adres → EEADR dane → EEDATA (EEPGD = 0) EECON1.WREN = 1 EECON2 = 55h EECON2 = AAh EECON1.WR = 1 (Δt ≈ 10 ms) EEIF = 1, EECON1.WR = 0

PIC (1)

- PIC16F8x – pamięć FLASH
  - Jak EEPROM, ale dłuższe rej. adresowe i danych
  - Zapis tylko do obszaru niezabezpieczonego

Read	Write
adres → EEADR, EEADRH EEPGD = 1 EECON1.RD = 1 nop *Δt dla odczytu z* nop *pamięci programu* EEDATA, EEDATH → dane	adres → EEADR, EEADRH dane → EEDATA, EEDATH (EEPGD = 1) EECON1.WREN = 1 EECON2 = 55h EECON2 = AAh EECON1.WR = 1 nop *µp pomija* nop *Δt ≈ 10 ms* EEIF = 1, EECON1.WR = 0

  - Programowanie zdalne z użyciem „*boot-loadera*”