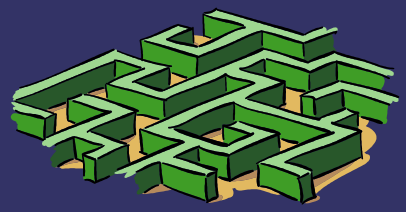
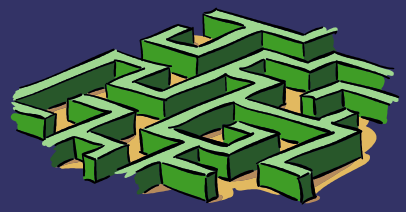
PICmicro™ assembler

Programmeerimiskeeled
Toomas Laasik



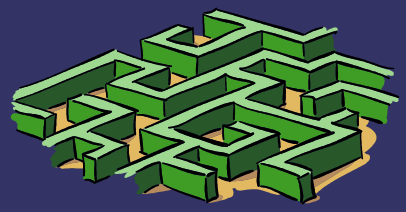
Ülevaade

- Loodud Microchip® mikrokontrollerite jaoks
- Riistvarast tugevasti sõltuv
- Tagasiühilduv
- Lihtne käsustik



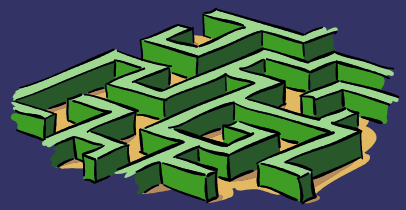
Kontrollerid

- 8 bitised RISC arhitektuuriga
- Programmi mälu 0.375KB - 128KB (256-65526 instruktsiooni)
- RAM 16B – 4KB
- Kiirus 4 – 50MHz
- Hind $0.39 – 7$
-
- Keel sobib üle 200 erineva kontrolleri jaoks



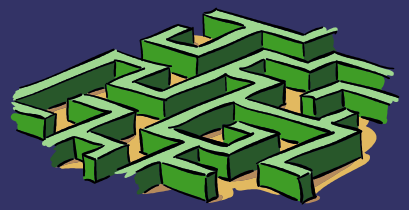
Kontrollerite perekonnad

- Base-Line
  - 12bit instruktsiooni sõnad
  - 1-5 MIPS
- Mid-Range
  - 14bit instrustiooni sõnad
  - 5 MIPS
- High-End
  - 16bit instuktsiooni sõnad
  - 10+ MIPS

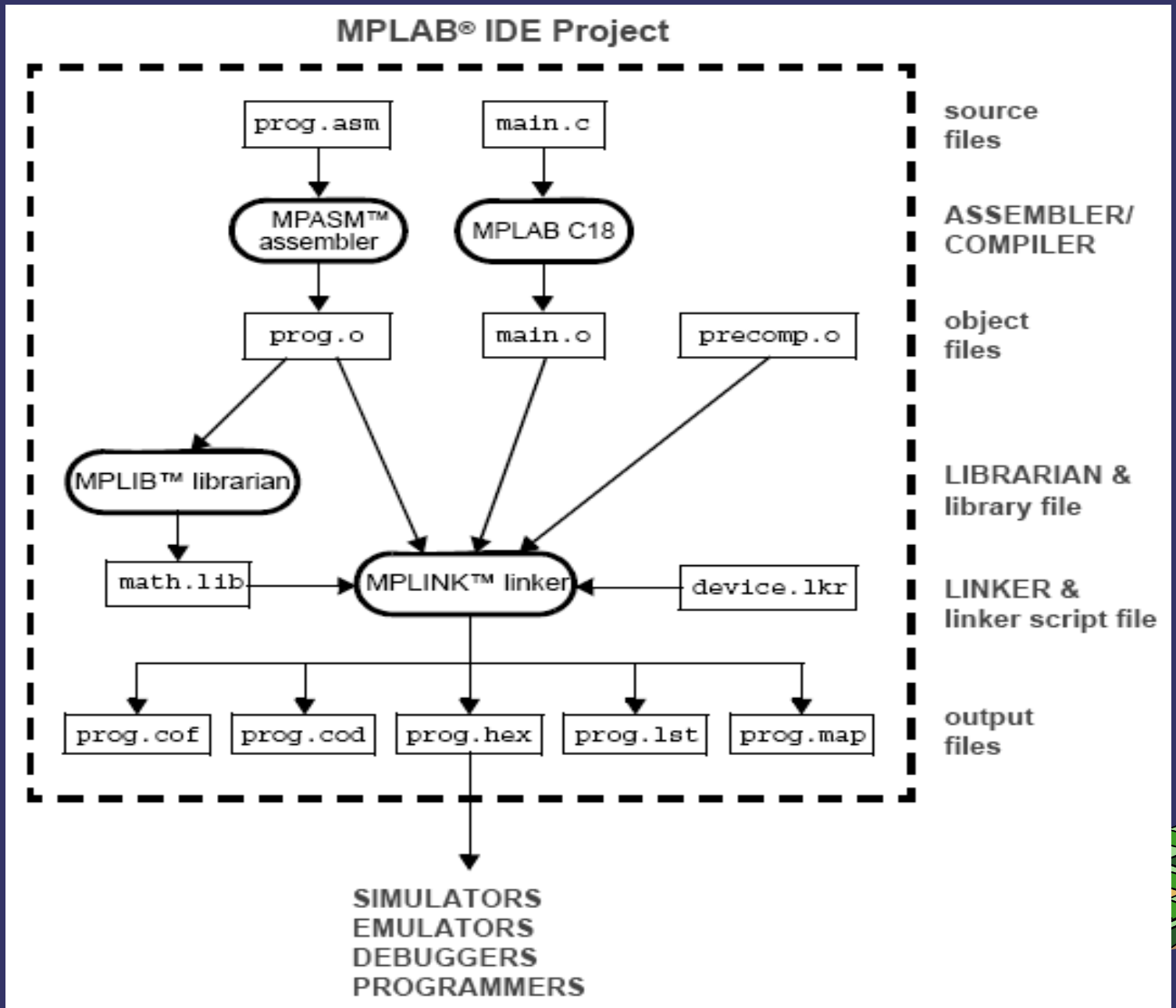


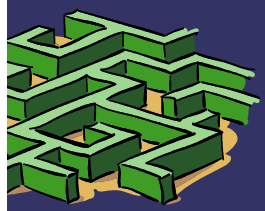
PICmicro assembler

- 33-75 erinevat operatsiooni vastavalt perekonnale
- Operatsioonide tüübid:
  - Bait-orienteeritud operatsioonid
  - Bit-orienteeritud operatsioonid
  - Literaalide ja kontroll operatsioonid
- Riistvarast tugevasti sõltuv
  - 1 akumulaatori register (W)
  - Failiregistrid (F)
    - Mälu
    - I/O
    - Erineva seadmed



Mis tehakse koodiga?

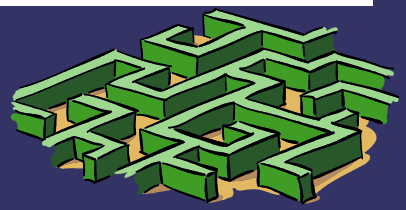




ASM fail

- Ühel real kuni 4 info tüüpi:
  - Label
  - Mnemonics, direc-tives, macros
  - Operands
  - Comments

```
                Mnemonics
                Directives
Labels          Macros          Operands        Comments
↓               ↓               ↓               ↓

                list            p=18f452
                #include        p18f452.inc

Dest            equ             0x0B            ;Define constant

                org             0x0000          ;Reset vector
                goto            Start

                org             0x0020          ;Begin program

Start
                movlw           0x0A
                movwf           Dest

                bcf             Dest, 3 ;This line uses 2 operands
                goto            Start
                end
```



Käsustik

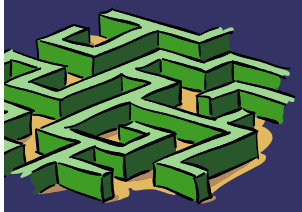
TABLE 15-2: PIC16F87XA INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add Literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND Literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call Subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to Address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR Literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move Literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from Interrupt	2	00 0000 0000 1001			
RETLW	k	Return with Literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into Standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from Literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR Literal with W	1	11 1010 kkkk kkkk		Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

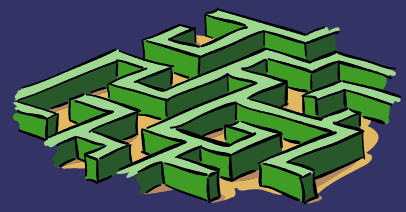
2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.



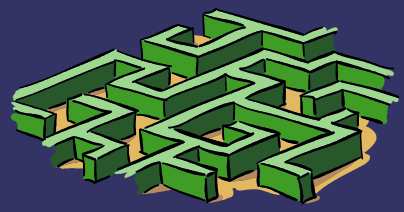
Näiteid koodist - IF

```
proc1   RETURN                ;proc(){return;}

        ...
        BTFSS PORTA, 3        ;if(A[3]=0)
        CALL  func1           ; proc1();

        ...
```



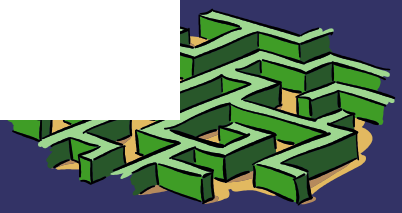
Näited koodist - IF2

```
#define _Z ALUSTA,2        ;defineerime registri

func1   RETLW 'A'          ;func1(){return 'A';}
        ...
        MOVLW 0x7A         ;W:=0x7A;
        XORWF PORTA        ;if(A==W)
        BTFSC _Z           ;
        CALL  func1        ; W:=func1();
        ...
```



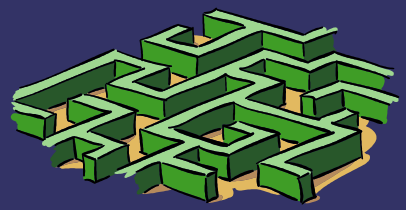
Näiteid koodist - WHILE

```
RAM0     equ      020h

proc1    MOVFW RAM0
         MOVWF PORTA
         RETURN
         ...
         MOVLW 10           ; ram0=10;
loop     DECFSZ RAM0        ; while(ram0>0) {
         GOTO loop_end      ;  ram0--;
         CALL proc1         ;  proc1();
         GOTO loop          ; }
loop_end ...
```



Arendusvahendid

- Riistvaraline programmaator (võimalik ka ise ehitada)
- MPLAB® IDE
  - Compiler
  - Editor
  - Simulator
  - In circuit debugger



Tänan kuulamast
Küsimused?