



Sulautetut prosessorijärjestelmät

(4 op)



Demot

Materiaali perustuu Microchipin ([www. microchip.com](http://www.microchip.com)) MPLAB IDE assembler-kääntäjän ja linkittäjän dokumentteihin ja PIC18F452 datalehteen.



18F452 käskykanta

- 76 käskyä
- Käskyt 16 bittisiä
 - Poikkeuksina MOVFF, LFSR, GOTO ja CALL ovat 32 bittisiä => vaativat 2 konejaksoa
- Käsky koostuu 4-8 bittisestä operaatiokoodista ja 8-12 bittisestä datasta
 - Data voi olla literal-arvo tai osoite
 - Yhteen käskyyn mahtuu vain yksi osoite
 - Paitsi MOVFF kaksi osoitetta



18F452 käskykanta

- Useat käskyt ovat muotoa:
 - MNEMONIC, f [, d [, a]]
 - MNEMONIC, k
 - MNEMONIC, [s]
 - MNEMONIC, f, b [, a]
 - f = 8 bittinen osoite rekistereihin (0x00 – 0xFF)
 - d = kohde (d='0' => WREG, d='1' => rekisterit)
 - a = access pankki (a='0' => access, a='1' => pankitettu)
 - [] –sisällä olevat argumentit ovat valinnaisia
 - k = literal-arvo (8, 12 tai 20 bittiä)
 - s = nopea kutsu / paluu valinta (s='0' => ei siirretä varjorekistereihin / -stä, s='1' => STATUS, WREG ja BSR siirretään varjorekistereihin / -stä)
 - b = bitin valinta f:n osoittamasta osoitteesta



18F452 käskykanta

- Voidaan jakaa viiteen kategoriaan:
 - Tavu-käskyt: operoivat tavu kerrallaan
 - XXXXX f [, d [,a]] ADDWF PORTB, W, ACCESS
 - Bitti-käskyt: operoivat bitti kerrallaan
 - XXXXX f, b [, a] BSF PIE1, ADIE, ACCESS
 - Literal-käskyt: käyttävät yhtä kiinteää operandia
 - XXXXX [i,] k MOVLW 0x12
 - Kontrolli-käskyt: ohjelman suorituksen ohjaus (hyppyt, kutsut, etc.)
 - XXXXX n [, s] BRA label
 - XXXXX f, b [, a] BTFSS PIE1, ADIE, ACCESS
 - XXXXX f [, d [, a]] INCFSZ PORTB, W, ACCESS
 - XXXXX k RETLW 0x35
 - Muut käskyt: PUSH, POP, SLEEP, RESET, NOP, CLRWDT, etc.



Tavu-käskyt

- Useat tavu-käskyt:
 - Operoivat mitä tahansa muistipaikkaa
 - $f=8$ bittinen vakio-osoite tai epäsuoraosoitus (INDFn)
Epäsuorassa osoituksessa osoite haetaan FSRn rekisteristä
 - Access-pankki tai pankitettu moodi ($a='0'$ tai $a='1'$)
Pankitetussa moodissa aktiivinen bank BSR rekisterissä
 - Tulos joko muistiin tai WREG:iin ($d='1'$ tai $d='0'$)



Bitti-käskyt

- Muokkaa / testaa yhtä bittiä
- Osoite f kuten tavu-käskyissä, lisäksi 3 bittiä joilla kerrotaan mitä bittiä muokataan (b)
 - BSF f, b [, a] Asetus $b \leq '1'$
 - BCF f, b [, a] Tyhjäys $b \leq '0'$
 - BTG f, b [, a] Vaihto $b \leq \text{not}(b)$
 - BTFSS f, b [, a] Asetettu? skip, if $b='1'$
 - BTFSC f, b [, a] Tyhjä? skip, if $b='0'$



Datan siirto

- **MOVLW** **k** Siirtää literalin WREG:iin, k 8 bittinen vakio luku
- **MOVLB** **k** Asettaa literalin BSR:ään, k 4 bittinen
- **LSFR** **i,k** Lataa FSRi:n, k 12 bittinen
- **MOVF** **f [, d [, a]]** Kopioi f:n sisällön
- **MOVFF** **fs, fd** Kopioi fs:n fd:hen
- **MOVWF** **f [, a]** Kopioi WREG:in f:ään



Lisäys / vähennys

- | | | |
|----------|-----------------|-------------------------------|
| • INCF | $f [, d [, a]]$ | Lisää f:ään yhden |
| • INCFSZ | $f [, d [, a]]$ | Kuin ed., skip jos $f=0$ |
| • INCSNZ | $f [, d [, a]]$ | Kuin ed., skip jos $f \neq 0$ |
| • DEC | $f [, d [, a]]$ | Vähentää f:ää yhdellä |
| • DECFSZ | $f [, d [, a]]$ | Kuin ed., skip jos $f=0$ |
| • DCFSNZ | $f [, d [, a]]$ | Kuin ed., skip jos $f \neq 0$ |



Aritmeettiset operaatiot

- NEGF $f [, a]$ $f \leq -f$ (2:n komplementti)
- ADDLW k $WREG \leq WREG + k$
- ADDWF $f [, d [, a]]$ $d \leq WREG + f$
- ADDWFC $f [, d [, a]]$ $d \leq WREG + f + \text{Carry}$
- DAW $WREG \leq \text{korjattu BCD luku}$
- SUBLW $f [, d [, a]]$ $WREG \leq k - WREG$
- SUBWF $f [, d [, a]]$ $d \leq f - WREG$
- SUBWFB $f [, d [, a]]$ $d \leq f - WREG - \text{not (Carry)}$
- SUBFWB $f [, d [, a]]$ $d \leq WREG - f - \text{not (Carry)}$



Aritmeettiset operaatiot

- `MULLW k PROD{H:L} <= k * WREG`
- `MULWF f[, a] PROD{H:L} <= f * WREG`
 - Molemmissa WREG ja f pysyvät muuttumattomina!
- Muistibittien (Carry tai Borrow) kanssa suoritettujen yhteen- ja vähennyslaskujen helpottaminen monitavusteisten numeroiden käsittelyä
 - Carry = not (Borrow) vähennyslaskussa
- Koska vähennyslaskun tulos riippuu järjestyksestä, on tarjolla 2 vaihtoehtoa



Loogiset operaatiot

- CLRF $f [, a]$ $f \leq 0$
- SETF $f [, a]$ $f \leq 0xff$
- Seuraavat komennot bittikohtaisia (bitwise)
- COMF $f [, d [, a]]$ $f \leq \text{not } (f)$
- ANDLW k $WREG \leq WREG \text{ and } k$
- ANDWF $f [, d [, a]]$ $d \leq WREG \text{ and } f$
- IORLW k $WREG \leq WREG \text{ or } k$
- IORWF $f [, d [, a]]$ $d \leq WREG \text{ or } f$
- XORLW k $WREG \leq WREG \text{ xor } k$
- XORWF $f [, d [, a]]$ $d \leq WREG \text{ xor } f$



Loogiset operaatiot

- Asetetaan bitit <3> ja <0>, muut eivät muutu
 - IORLW '00001001'
- Tyhjätään bitit <2> ja <1>, muut eivät muutu
 - ANDLW '11111001'
- Muutetaan bitit <7> ja <6>, muut eivät muutu
 - XORLW '11000000'



Pyörityskäskyt

- RLCF $f [, d [, a]]$ Pyörittää vasempaan,
Carry:n kautta
- RLNCF $f [, d [, a]]$ Kuin ed., ilman Carry:ä
- RRCF $f [, d [, a]]$ Pyörittää oikeaan,
Carry:n kautta
- RRNCF $f [, d [, a]]$ Kuin ed., ilman Carry:ä
- SWAPF $f [, d [, a]]$ $d \leq f$:n ylimmät 4 bittiä
vaihtavat paikka
alimman 4:n kanssa
- Vasemmalle pyöritys = kertominen kahdella
- Oikealle pyöritys = jakaminen kahdella



Vertailuja

- TSTFSZ $f [, a]$ skip, jos $f = 0$
- CPFSEQ $f [, a]$ skip, jos $f = \text{WREG}$
- CPFSGT $f [, a]$ skip, jos $f > \text{WREG}$
- CPFSLT $f [, a]$ skip, jos $f < \text{WREG}$
 - CPFSGT ja CPFSLT ovat etumerkittömiä, koko tavun mittaisia vertailuja



Hyppykäskyt

- BRA n Ehdoton hyppy (± 512 käskyä)
- BC n Hyppy, jos Carry = '1'
- BNC n Hyppy, jos Carry = '0'
- BN n Hyppy, jos Negativinen (N)
- BNN n Hyppy, jos ei Negatiivinen
- BOV n Hyppy, jos ylivuoto (OV)
- BNOV n Hyppy, jos ei ylivuotoa
- BZ n Hyppy, jos nolla (Z)
- BNZ n Hyppy, jos ei nolla
- n = ± 64 yhden sanan käskyä, Assembler laskee labeleista
- Testattavat bitit STATUS-rekisterissä



Kutsut ja paluut

- GOTO n Ehdoton hyppy (koko ohjelma muisti)
- CALL n [, s] Aliohjelma kutsu (koko ohjelma muisti)
- RCALL n Aliohjelma kutsu (± 512 käskyä)
- RETURN [s] Paluu aliohjelmasta
- RETLW k Paluu aliohjelmasta, WREG \leq k
- RETFIE [s] Paluu keskeytyksestä
- n:t labeleitä, Assembler-kääntäjä laskee todelliset ohjelmamuistin osoitteet



Muut käskyt

- PUSH PC talletetaan paluosoitepinoon
- POP Paluosoitepinon ylin hävitetään
- CLRWDT Nollaa WatchDogin
- SLEEP Menee Standby-moodiin (ja nollaa WatchDogin)
- RESET SW reset, sama kuin MCLR-pinnillä
- NOP Ei mitään (No OPeration)
- TBLRD Taulukon luku (ohjelmamuisti)
- TBLWT Taulukon kirjoitus (ohjelmamuisti)



Aiempien vuosien demoissa havaittua

- BRA \neq GOTO

- BRA voi hypätä ± 512 käskyn päähän nykyisestä rivistä
 - Luennoilla jaetussa 2-sivuisessa käskykannassa virhe!
 - Siinä väitetään että ± 64 käskyä
- GOTO voi hypätä koko ohjelmamuistiin
- BRA vie 1 paikan ohjelmamuistista
- GOTO vie 2 paikkaa
 - Eli myös suoritusnopeudessa on eroa



Assembler

- Minimissään muuntaa assembly *muistikkaat* binäärikoodiksi
 - ADDLW 0xAA => 00001111 10101010
 - (MOT Enteka 3.0a sanakirja kääntää seuraavasti:
mnemonic: mnemoninen, muistikas (atk))
- Myös osoitteiden laskenta automaattisesti
 - Labelit koodin joukossa saavat numeroarvon
 - Myös suhteelliset osoitteet
 - Datamuistiinkin voidaan yhdistää nimettyjä osoitteita (= nimettyjä muuttujia)



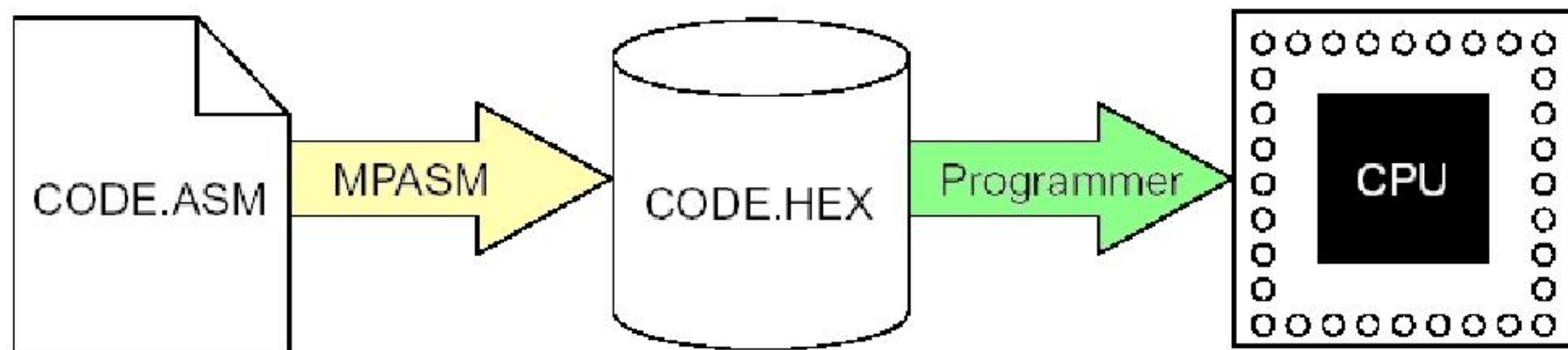
Assembler

- Yleensä myös:
 - Komentoja käännöksen ohjaukseen
 - Esim. ehdollinen käännös, valitaan tietty koodin pätkä pienemmälle PIC:ille ja toinen isommalle
 - => sama lähdekoodi voidaan kääntää useammalle eri laitteelle
 - Makro-laajennukset
 - Include-tiedosto
 - Käskykannan ”laajentaminen”
 - Korkeampi abstraktiotaso
 - Laskentaa **käännöksen** aikana
 - Voidaan antaa vakioita, joista lasketaan esim. lämpötilan kalibrointi




Kääntäminen

- Helppoa, jos vain yksi lähdekooditiedosto
 - Useampia varten tarvitaan erillinen linkkausvaihe, jonka aikana osoitteet lasketaan uudestaan ja sovitetaan toisiinsa
 - Esim. esikäännettyjen kirjastojen käyttö






MPASM (-WIN)

 **MPASM v03.60 - Microchip Technology, Inc.**

Source File Name:


MICROCHIP

Options:

<p>Radix:</p> <p><input checked="" type="radio"/> Default</p> <p><input type="radio"/> Hexadecimal</p> <p><input type="radio"/> Decimal</p> <p><input type="radio"/> Octal</p>	<p>Warning Level:</p> <p><input checked="" type="radio"/> Default</p> <p><input type="radio"/> All Messages</p> <p><input type="radio"/> Warnings and Errors</p> <p><input type="radio"/> Errors Only</p>	<p>Hex Output:</p> <p><input checked="" type="radio"/> Default</p> <p><input type="radio"/> INHX8M</p> <p><input type="radio"/> INHX8S</p> <p><input type="radio"/> INHX32</p>	<p>Generated Files:</p> <p><input checked="" type="checkbox"/> Error File</p> <p><input type="checkbox"/> List File</p> <p><input type="checkbox"/> Cross Reference File</p> <p><input type="checkbox"/> Object File</p>
<p><input checked="" type="checkbox"/> Case Sensitive</p> <p>Tab Size: <input type="text" value="8"/></p>	<p>Macro Expansion:</p> <p><input checked="" type="radio"/> Default</p> <p><input type="radio"/> On</p> <p><input type="radio"/> Off</p>	<p>Processor: <input type="text" value="18F452"/></p> <p><input type="checkbox"/> Extended Mode</p>	

Extra Options:

☒ Save Settings on Exit



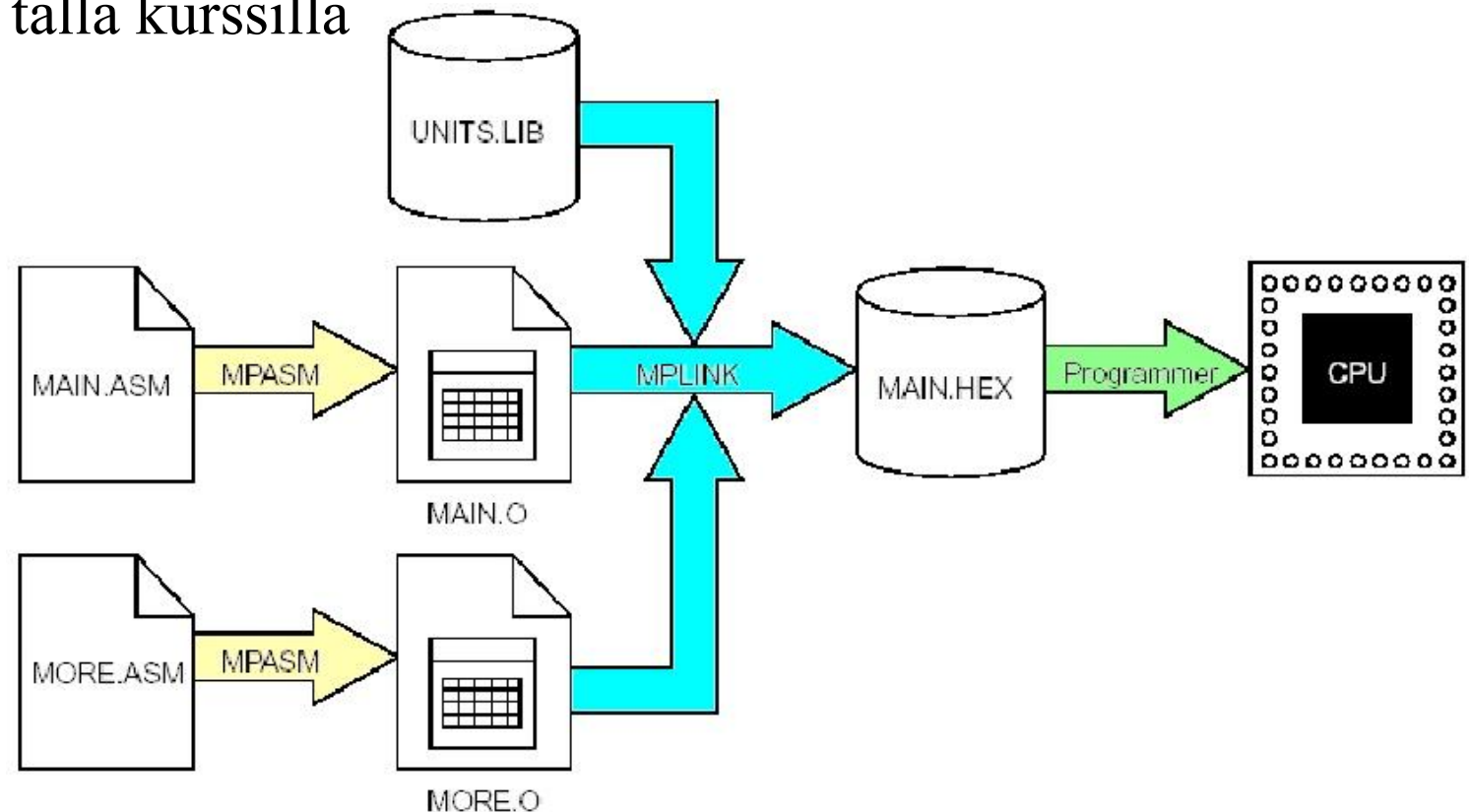
MPASM (-WIN)

- Ottaa sisään lähdekoodin, yleensä .ASM
- Tuottaa seuraavat tiedostot:
 - .HEX Binäärikoodi, absoluuttiset osoitteet, käännöksen tulos
 - .LST Listaa lähdekoodin ja mukaan otetut include-tiedostot, ja näyttää jokaista riviä vastaavan käännöksen
 - .ERR Käännöksessä sattuneet virheet
 - .XRF Viittaukset (osoitteet, vakiot, makrot...)
 - .O Objektitiedosto, linkkausta varten, osoitteet muutettavissa



Linkkeri

- Ensin käännökset assemblerilla, sitten objektitiedostot linkataan
 - Lasketaan absoluuttiset osoitteet
 - MPLINK
 - Ei tarvita tällä kurssilla





Lähdekoodin rakenne

- Rivillä voi olla:
 - Nimiö (label) = hyppyosoite tai määriteltävä termi
 - Muistikas (mnemonic) = käsky
 - Operandi (-t) = vakioluku / osoite
 - Kommentteja
 - Kääntäjän ohjaamiseen käytettyjä käskyjä = direktiivit
 - Korkeintaan 255 merkkiä / rivi



Lähdekoodin rakenne

- Nimiöt (=hyppyosoitteet, määriteltävät termit)
 - Alkavat sarakkeesta 1 eli ensimmäinen merkki vasemmalta
 - Isot ja pienet kirjaimet merkittäviä (case-sensitive)
 - Pitää alkaa kirjaimella tai alaviivalla (_)
 - Maksimipituus 32 merkkiä
- Muistikkaat (= käskyt)
 - Ei saa alkaa sarakkeesta 1
- Operandit (=muistiosoitteet, bitin järjestysnumero, vakiot)
 - Muistikkaan perässä
 - Erotetaan pilkulla, jos useita
- Kommentit
 - Voivat alkaa mistä vaan, alkaa puolipisteellä (;)
 - Vaikuttaa rivin loppuun asti



Lähdekoodin rakenne

- Käännöksen ohjaus:
 - Assembler komentoja (direktiivit) voidaan sijoittaa koodin joukkoon kertomaan miten / mitä käännetään
 - Valinnat tapahtuvat käännöksen aikana, ei ohjelman suorituksessa
 - EQU: määrittää vakion
 - <nimiö> equ <arvo>
 - Arvo on käytössä käännöksen aikana valintojen tekemiseen tai datana käskyille. Käännöksen alussa <nimiö> korvataan <arvo>:lla tekstinä.
 - Ei ohjelman suorituksessa muutetavana muuttujana
 - SET: määrittää muuttujan (ei suoritusaikainen)
 - <nimiö> set <arvo>
 - Voidaan muuttaa uudella set-käskyllä
 - Käännöksen aikana annetut arvot vakioita ohjelmaa suoritettaessa
 - Siis ei myöskään ohjelmaa suoritettaessa muutettava muuttuja



Lähdekoodin rakenne

- Käännöksen ohjaus:
 - ORG: sijoittaa koodin alkamaan tästä eteenpäin tietystä osoitteesta
 - [<nimiö>] org <arvo>
 - Esim. org 0x000008 seuraavat rivit päätyvät keskeytysvektoriksi
 - #DEFINE: määrittää tekstisymbolin
 - #define <nimiö> [= <merkkijono>]
 - #UNDEFINE: poistaa edellisen vaikutuksen
 - #undef <nimiö>
 - IFDEF: lohko käännetään jos <nimiö> on määritetty
 - IFNDEF: lohko käännetään jos <nimiö> ei ole määritetty
 - ELSE
 - ENDIF: päättää ehdollisen lohkon
 - ifdef <nimiö>
 <pätkä koodia>
endif



Lähdekoodin rakenne

- Käännöksen ohjaus:
 - #INCLUDE: liittää tähän kohtaan toisen tiedoston
 - #include "polku\tiedosto" tai #include <polku\tiedosto>
 - Toimii kuten ANSI C kääntäjässä
 - "tiedosto" ja <tiedosto> käyttäytyvät samoin
 - Tiedostoja haetaan:
 - Nykyisestä hakemistosta
 - Lähdekoodin hakemistosta
 - MPASM:in omasta hakemistosta
 - Mahdollistaa kirjastojen käytön ilman erillistä linkkausta
 - Aliohjelmia erillisissä tiedostoissa jotka otetaan mukaan includella kuin ne olisi kirjoitettu kaikki samaan tiedostoon
 - Ei kuitenkaan voida käyttää esikäännetyille kirjastoille



Include-tiedosto

- Sisältää ennalta määriteltäviä literaaleja
 - Samoja symboleja käytetty luennoissa / datasheeteissä
 - Koodista tulee huomattavasti luettavampaa
 - Debuggaus helpottuu
 - Tarkastaminen helpottuu
- Esimerkiksi:
 - `ADDWF H'81', 0, 0`
 - `ADDWF PORTB, W, ACCESS`
 - Tekevät saman asian (laskee yhteen portin B sisältämän datan ja WREG:in, tulos WREG:iin) mutta jälkimmäinen luettavampi
- Tiedoston nimi P18F452.INC
 - Tarjolla kurssin kotisivulla
 - Kannattaa tutustua viimeistään ennen 3. demoja
 - EI OPETELLA ULKOA



Lähdekoodin rakenne

- Listauksen ohjaus (.LST):
 - TITLE: otsikko
 - SUBTITLE: aliotsikko
 - title ”otsikko”
 - subtitle ”aliotsikko”
 - Tulostuvat jokaiselle sivulle .LST –tiedostoon
 - SPACE: lisää .LST –tiedostoon tyhjän rivin
 - PAGE: aloittaa uuden sivun .LST –tiedostossa
 - LIST, NOLIST: kytkee listauksen päälle / pois
 - EXPAND, NOEXPAND: kytkee macrojen laajennuksen päälle / pois
 - MESSAGE: kirjoittaa viestin / kommentin



Lähdekoodin rakenne

- Numerojärjestelmät: (suluissa kantaluku)
 - Hexadesimaali (16) H'A3' tai 0xA3 tai 0A3h
 - Desimaali (10) D'163'
 - Oktaali (8) O'243'
 - Binääri (2) B'10100011'
 - Oletusarvona hexadesimaali
 - ellei muutettu lähdekoodissa tai kääntäjän käyttöliittymässä
 - ASCII merkit 'C' tai A'C'



Lähdekoodin rakenne

- Kääntäjän konfigurointi:
 - RADIX: asettaa oletuksena käytettävän numerojärjestelmän
 - hex, dec tai oct
 - Oletuksena hex
 - ERRORLEVEL: säättää ilmoitettavien virheiden määrää / laatua
 - 0 => kaikki virheet ja varoitukset ilmoitetaan
 - 2 => mitään ei ilmoiteta
 - Käytetään aina 0 –tasoa
- Käännöksen lopetus:
 - END: lopettaa käännöksen
 - Pitää olla ohjelman lopussa (aina)



Lähdekoodin rakenne

EXAMPLE: ABSOLUTE MPASM ASSEMBLER SOURCE CODE (SHOWS MULTIPLE OPERANDS)

Labels	Mnemonics Directives 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		



Lähdekoodin rakenne

- Ehdollinen käänнос:
 - Jo aiemmin esitelty #DEFINE –metodi
 - IF, ELSE, ENDIF: ehdon perusteella valitaan käännetäänkö jotain lohkoa

```
#define p18F452
#ifdef p18F452
    movlw    0x05
    movwf    PORTB
else
    clrf     PORTB
#endif
```



Lähdekoodin rakenne

- Viisi erilaista END:iä
 - END ohjelman loppuun
 - ENDIF ehdollisen lohkon loppuun
 - ENDW while-loopin loppuun
 - ENDM macron loppuun
 - ENDC vakiolohkon loppuun

Kaikki nämä ovat käännoisaikaisiin toimintoihin liittyviä