System software

Assembly language

Assembler

- Command types
  - instructions
    - directly represent instructions supported by a processor
  - directives
    - command which guide the compilation (assembly) process
- Mnemonics
  - symbolic names for instructions and directives

Assembler

- Operands
  - instructions & directives can have operands
  - SIC/XE operand kinds
    - format F1
    - format F2
    - format F2
    - format F3 … extended instructions

Assembler

- Operand kinds
  - format F1: no operands
    - `RSUB`
    - `NOBASE`
  - format F2: r – register A, S, T, L, B, X
    - `ADDR  A, S`
    - `CLEAR  T`

Assembler

- Format F3 & F4: addressing – use of TA
  - simple – operand is at the given address
    - operand without prefix
    - `LDA  42`
  - immediate – operand is part of the opcode
    - prefix #
    - `LDA  #42`
  - indirect – address of the operand is at the given address
    - prefix @
    - `LDA  @42`

Assembler

- Format F3 & F4: operand kinds
  - s – symbol
    - `BASE    podatki`
    - `LDA     count`
    - `J       halt`
  - * – current address (LOCCTR)

Assembler

- Format F3 & F4: operand kinds
  - n – number (extension)
    - `LDA  4095  ; 12-bit unsigned number`
    - `+LDA  1048575  ; 20-bitno unsigned number`
    - `LDA  0xFFF`
    - `LDA  0o7777`
    - `LDA  0b101010101010`
    - immediate behaves a bit differently
    - `LDA  #-1  ; 12 bit signed number`

Assembler

- Format F3 & F4: operand kinds
  - =literal
    - literal is of the same form as with `BYTE` / `WORD`
  - What is the difference?
    - `LDA  123`
    - `LDA  @123`
    - `LDA  #123`
    - `LDA  =123`

Assembler

- Directives
  - *name* `START` *address*
    - program name
    - origin address – starting address of the program
  - `END` *first*
    - address where the execution of the program begins
  - `ORG` *address*
    - new translation address

Assembler

- Directives
  - `BASE` *address*
    - enables base addressing
    - tells assembler the value of the register `B`
    - first we load `B`, then we enable base addressing
      - `+LDB #podatki`
      - `BASE podatki`
  - `NOBASE`
    - disables base addressing

Assembler

- Symbols
  - *name* `EQU` *expression*
    - assignes the value of expression to name
      name = expression
  - `EXTDEF` s,...
  - `EXTREF` s,...
    - export and import of symbols
  - `LTORG`
    - flush not yet flushed literals

Assembler

- Sections and blocks
  - *name* `CSECT`
    - consecutive block of code (or data)
    - name is automatically exported
  - `USE` *name*
    - enters into the given block
    - without the parameter, re-enters the previous block

Assembler

- Memory reservation
  - `RESB` *number*
    - reservers the given number of bytes
  - `RESW` *number*
    - reservers the given number of words
  - `RESF` *number*
    - reservers the given number of floats
    - extension

Assembler

- Memory reservation
  - `BYTE` *data*
    - initializes memory with the data
  - `WORD` *data*
    - initializes memory with the data
    - rounds the memory size to a multiple of 3
  - `FLOT` *data*
    - initializes memory with the data
    - rounds the memory size to a multiple of 6

Assembler

- Memory reservation
  - `C'chars...'` – characters
    - `BYTE C'spo'`
    - `WORD C'great'` ... initializes 6 bytes, zero padding
  - `X'hex...'` – hex encoding
    - `BYTE X'FFAA001A2B'` ... initializes 5 bytes
    - `WORD X'BAD123'` ... inictializes 3 bytes
  - number
    - extension
    - 8 or 24 bit number
    - signed or unsigned