Prometheus

Z80 Turbo assembler

Prometheus is not only assembler, it has powerfull excellent monitor, it's better to call it "developing and debugging system".

History:
Prometheus was developed using GENS 3.1 and GENS 3E. Source was about 40 kB long, divided into two parts, plus about 5kB label table each part. Assembly time was 40 s.
Then source was converted to first Prometheus version and assembler was developed on itself! After converting was source lenght 20 kB (with label table, it's integral part of source), assembly time was 3 s.

Versions:
Prometheus 48
Prometheus: 11kB assembler, 5kB monitor, relocable, independent on system variables (can be placed at 23296), source/assembled code ratio 3.5/1 (GENS 8/1), compilation 3kB/sec, supports tape, monitor can measure tacts taken by routine. Probably the best assembler for ZX 48 with tape!
DISKPROM 48, DISKASSEM 48: relocable 48 version, supports tape or D40/80 disk (MDOS). DISKPROM is with monitor (16 kB), DISKASSEM without monitor (11kB). Warning, D40 needs system variables!
TR-DOS version of Prometheus 48 is available somewhere, too...
Prometheus 128
upto 64kB source, assembly is a bit slower than 48 version.
SHT (short, 17500 bytes): relocable, supports D40 only, some functions were removed, texts were shortened...
MDM (medium, 18589 bytes): relocable, supports tape and D40, good to create 48k progs, can't trace 128k prog.
LNG (long, 19122 bytes): relocable, supports tape and D40, can do everything (assembly to pages, trace 128 progs, ...)
Prometheus 128 LNG for MB-02+: Non relocable (two versions: at 24576 and at 45920), supports tape and M-02+ (BS DOS). Adapted by Jordan of Exodus/Reaction.

Installation:
Relocable versions:
Load Prometheus to any address. Then **Instalation address:**xxxxx appears (xxxx is actual start address). You can change it using number keys and delete.
(48 kB only):
On your screen are displayed several things, as **Monitor:Yes** text. You can change it using **M** key. To set colours use: **P** (paper), **I** (ink), **B** (bright), with **CAPSSHIFT** for highlighted line. To change font press **D**. To set keyboard echo lenght, press **X** or **CS+X**. Press **C** to change texts (usually Prometheus displays instructions lowcased and labels and constants upcased; you can change it to all letters upcased or lowcased).
After installation press **Enter**.

Assembler:

Editor:
On screen you can see:
```
Error message       Insert/Overwrite    Source end   U-top
            Source text
 highlighted line (in source text)
            Source text
   Edit zone - here you can edit the highlighted line
```

(Source end in 48 version is real end, in 128 0 means 1st byte of page 1, 65535 last byte of page 6. Source is stored (usually, you can change it) in pages 1, 3, 4, 6.)

Control keys:
CS+1 (edit) - edit highlighted line.
CS+2 (capslock) - caps lock (inverts function of capsshift, too).
CS+3 (true video) - page down.
CS+4 (inv video) - page up.
CS+5 (left) - moves cursor in edit zone left.
CS+6 (right) - moves cursor in edit zone right.
CS+7 (up) - moves highlighted line up.
CS+8 (down) - moves highlighted line down.
CS+9 (graph) - delete line.
CS+0 (delete) - backspace in edit zone.
SS+Q - clears edit zone.
SS+W - switches Insert/Overwrite.
SS+H - switch HEX / DEC (in info line and symbol table).

Editor commands:
You can enter a command only if the edit zone is empty. Then press symbolshift+key, a command appears, then you can type parameters.
These two commands are executed after pressing command key:
SS+E - go to text end.
SS+K (+) - go to text begin.

Block commands:
To select block press **CS+SS**.
COPY (SS+C) - copy block.
COPY m - move block (as copy and delete), 128 version only.
DELETE (SS+D) - delete block.
Search commands:
FIND (SS+F)
FIND S:XXX - find "XXX" from beginning.
FIND s - find next from beginning.
FIND b:XXX - find "XXX" in block from block begin.
FIND:XXX - find "XXX" from highlighted line to end of text or block (if the last used FIND was with b parameter).
FIND - find next fromhighlighted line to end of text or block.
(Example: to skip at block begin type:"**FIND b:** " (one space).)
REPLACE (SS+Z)
REPLACE:YYY - replaces all "XXX" (found by last FIND) in highlighted line with "YYY", and finds next "XXX".
REPLACE - replaces äll "XXX" (found by last FIND) in highlighted line with "YYY" (used by last REPLACE), and finds next "XXX".

I/O commands:
CLEAR (SS+X)
CLEAR y - clears source, only locked labels survive. Because it's dangerous function, this Y as YES is necessary for confirmation.
CLEAR f - in 128 version only, fast clear. Clears all, it means locked labels too!
U-TOP (SS+U) - in 48 version, end of place for source and compiled code. In 128 version, it's end of place for compiled code.
S-BEGIN number (SS+I) - 128 only, set begin of source (0 is 1st byte of page 1, 65535 is last byte of page 6).
S-TOP(SS+J) - 128 only, displays S-BEGIN and S-TOP.
S-TOP number - 128 only, set end of place for source (0 is 1st byte of page 1, 65535 is last byte of page 6).
SAVE:name (SS+S) - saves source and symbol table.
SAVE - saves source and symbol table as last used name.
SAVE b:name, SAVE b - saves block (with full symbol table).
SAVE t, SAVE d - 128 only, selects disk or tape for all I/O operations.
SAVE a - 128 only, D40 specific. Switches drives A: and B:.
LOAD:name (SS+L) - load file. Warning, Prometheus performs only MERGE, not real LOAD; real LOAD can only 128 version, if no source is typed in! After merge (LOAD) is source line by line inserted (like from keyboard) (you can display it using any key or edit some line using SPACE).
LOAD: - loads first source (tape only).
LOAD - loads file with last used name (SAVE or LOAD commands).
LOAD:! - with D40 performs CATalogue of disk.
VERIFY (SS+V) - verify on tape, uses last used name (after SAVE).
GENS (SS+G) - syntax as LOAD command. Imports file from GENS assembler to Prometheus (GENS source format: two bytes line number, text (space for tabelation), code 13 as line end). Not availale in MB-02+ version.
PATCH (SS+G) - performs CATalogue of disk (MB only).
PATCH $0, PATCH @1, PATCH @1$0 - sets path and performs CAT (MB only).

Other commands:
PRINT (SS+P) - prints via #3.
PRINT b - prints block, **TABLE p** - prints symbol table.
MONITOR (SS+M) - go to monitor (if installed).
MONITOR a - ASSEMBLY + MONITOR, assembles source first.
BASIC (SS+B) - return to basic, if system variables are corrupted by your program, use next command.
NEW (SS+N) - NEW emulation - clears memory upto RAMTOP and restores system variables.
QUIT y (SS+Q) - reset (jp 0), as dangerous function needs Y parameter as confirmation (YES).
CALC number (SS+Y) - calculates expression or label value (128 only).

Assembly commands:
ASSEMBLY (SS+A) - assembles.
ASSEMBLY b - assembles block.
RUN (SS+R) - assembles and runs code (if you remove some bugs, it automatically runs new version).
TABLE (SS+T) - shows symbol table. After one screen waits for key - SPACE to cancel, any other key to next page.
TABLE:text - 128 only, shows symbol table from the one beginning with "text" in alphabetical order (!) (example: labels are ABR, GAP, GUN, ZZOOM. TABLE:GB shows labels GUN, ZOOM).
TABLE p - prints symbol table via #3.
TABLE c - clear table. Removes unused labels from table, clears values of labels (saves locked labels, they aren't removed and their value is not cleared).
TABLE l - locks all labels.
TABLE u - unlocks all labels. It's good for divided compilation - you can write one part of source (example:ORG 25000:OP equ 40000:START ld bc,OP:ret), compile it, lock labels and clear source (labels are still stored). Then you can write second part (example:ORG OP:CALL START:RET). Its sense was on 48kB Spectrum, when it was necessary to divide long source.

Error messages while editing:
Bad mnemonic - nonexisting instruction.
Bad operand - bad operand.
Big number - larger than 65535.
Syntax horror - something is wrong here in Denmark...
Bad string - bad string in DEFB or DEFM instruction.
Bad instruction - mnemonic is good, but this instruction cannot have these operands.
Memory full - memory full (you surely have 48kB Spectrum).
XXXX unknown - nondefined label used, for example in U-TOP or CALC commands...
Source ERROR - corrupted file was loaded.

Error messages while assembling:
Bad PUT (ORG) - you tried to overwrite assembler, source or protected memory place.
XXXX unknown - nondefined label used.
Already defined - the same name of label used more time.
Big number - 8bit number (ld a,N), relative jump, (ix+N) overload.
ENT ? - you tried to RUN source without entry point.

Assembler instructions:
Expressions:
+,-,*,/ as usually, ? is MOD, $ is address counter, # means HEX, % means BIN, "text" is usual text, 'text' is inverted text ("tex","t"+128). Prometheus computes from left to right!

Z-80 instructions:
Prometheus can recognise all "known" and some "secret" Z80 instructions. In some instructions, working with address, you can write for example: ld a,(label. Then press enter and the character ")" appears. Warning - ld a,(hl is then ld a,(HL) as label, not register!
The "secret" ones are:
SLIA (shift left inverted arithmetic)
slia b c d e h l (hl) a
IX and IY halves
inc hx lx hy ly
dec hx lx hy ly
ld hx lx hy ly,N
ld register,hx lx hy ly
ld hx,hx hx,lx lx,hx lx,lx
ld hy,hy hy,ly ly,hy ly,ly
add a,hx lx hy ly
adc a,hx lx hy ly
sub hx lx hy ly
sbc a,hx lx hy ly
and hx lx hy ly
xor hx lx hy ly
or hx lx hy ly
cp hx lx hy ly

Assembler pseudoinstructions:
Comment must begin with ";" at the begin of line (and takes all the line).
ORG number - address where final code will work
PUT number - address to store assembled code (example - screen located code: ORG 16384: PUT 60000. You'll then save code from 60000 and load to screen and run it).
PUT number - if the number is 0, 1, 3, 4, 6 or 7, code'll be assembled to that page of 128kB memory. 128 only.
Label **EQU** number - defines constant.
ENT number - defines entry point for RUN (usually ENT $ directly at the place).
DEFB number, number, number - byte values.
DEFW number, number, number - word values (two bytes).
DEFM text - "text" is normal text, 'text' is inverted (as "tex","t"+128).
DEFS number - leaves space (in bytes).

Internal source format:
Prometheus assembles line after you press ENTER while editing (so it provides syntax control) and in "source" are stored instructions as their real values - so source is NOT tokenized, but pre-assembled!

First byte:
Operation code of instruction or pseudoinstruction (pseudoinstructions have in second byte nonexisting combination of prefixes).
Second byte = Information byte:
bit 7 - prefix #CB (203)
bit 6 - prefix #ED (237)
bit 5 - prefix #DD (221)
bit 4 - prefix #FD (251)
bit 3 - label used in this instruction
bit 2-0 - type: 0 - without operand
                 1 - 1 byte, -256..256
                 2 - 2bytes, -65535..65535
                 3 - 1 byte, -128..128
                 4 - 1 byte, (ix+N)
                 5 - (ix+N),N
                 6 - rst p (p is stored in operation code)
                 7 - noninstruction
If bit 3=1 and bits 2-0 aren't only zero, third byte follows:
Bit 3=1, bits 2-0 =0: follows number of label, at the end 192+lenght.
Bit 3=0, bits 2-0 aren't only zero: follows all expression, labels as their numbers, at the end 192+lenght.
Bit 3=1, bits 2-0 aren't only zero: follows two bytes of label number, then expression, labels as their numbers, at the end 192+lenght.
Example:
2*LABEL+#23
"2","*","128+H,L,"+","#","2","3",192+8
Empty line has infobyte 48 or 56 (if label used), noninstructions have infobyte 55 or 63 (if label used). Opcode (first byte) of noninstructions:
                 0 - empty line
                 1 - comment
                 2 - ent
                 3 - equ
                 4 - org
                 5 - put
                 6 - defb
                 7 - defm
                 8 - defs
                 9 - defw
...it's very complicated.
Symbols table format:
It's simple:
Two bytes: total labels (counter)
2 bytes*counter: relation to names table (bits 6 and 7:locked and defined label)
Names table: two bytes value+'name' (inverted text)
All is stored in alphabetical order.

Monitor:

Prometheus monitor is very powerfull tool, a bit simillar to DevastAce or VAST monitors.

Memory:
Q - return to assembler
M - set new actual address. Question **Memory** appears. You can enter any numbe, label or expression (simply "number"). To cancel press **EDIT**.
A - 128 LNG only, switches memory pages (0,1,3,4,6,7).
B - 128 LNG only, turns memory switching on / off.
CS+6 - go to next instruction.
ENTER - one byte forward.
CS+7 - one byte back.
CS+8 - change depth - level down. Question **Memory** appears. You can go 10 levels down to program.
CS+9 - 128 only, change depth - level down. A value in actual instruction (for example CALL) is used.
CS+5 - change depth - level up. This returns you back to "surface".
CS+0 - clear screen.
CS+2 - clears only panels for dump and disassem.
SS+4 - disassembly to panel from actual address. Press **EDIT** to stop.
V - disassembly to panel. Question **First** appears. Press **EDIT** to stop.
SS+C - switch these modes of address display: number only / labels / label and label+1.
C - display address number / labels only.
SS+3 - switch HEX / DEC.
D - disassembly to printer via #3. Monitor asks for **First** and **Last**, the last will be NOT disassembled! Press CS+ENTER to stop.
SS+D - disassembly to source text. Monitor asks for **First** and **Last**, the last will be NOT disassembled! Press CS+ENTER to stop or EDIT to cancel.
O - text dump from actual address. Press **EDIT** to stop.
SS+O - text dump. Monitor asks you for **First** address. Press **EDIT** to stop.
L - dump from actual address. Press **EDIT** to stop.
SS+L - dump. Monitor asks you for **First** address. Press **EDIT** to stop.
G - find. Monitor asks you for 5 bytes (characters, numbers, expressions). For "any value" enter ":".
N - find next.
I - copy block. Monitor asks you for **First**, **Last** and target address **To**. READ/WRITE error means, that assembler, source or protected area is on target place.
SS+I - copy block. Monitor asks you for **First**, **Lenght** and target address **To**. READ/WRITE error means, that assembler, source or protected area is on target place.
P - fill block. Monitor asks you for **First**, **Last** and **With**. READ/WRITE error means, that assembler, source or protected area is on target place.
SS+P - fill block. Monitor asks you for **First**, **Lenght** and **With**. READ/WRITE error means, that assembler, source or protected area is on target place.
SPACE - edit one assembler line. You can write everything like in assembler (instructions, DEFB, DEFW, DEFM, EQU and define new labels, for example for disassembly). Press **EDIT** to cancel. Bad PUT(ORG) error means, that assembler, source or protected area is on target place.
E - edit more assembler lines. Press **EDIT** to cancel.

I/O operations:
S - save. Monitor asks you for **First** and **Last** byte and **Leader**. Leader can be value or **:name** (don't forget ":").
SS+S - save. Monitor asks you for **First** byte, **Lenght** and **Leader**. Leader can be value or **:name**.
J - load. Monitor asks you for **First** and **Last** byte and **Leader**. Leader can be value or **:name**.
SS+J - load. Monitor asks you for **First** byte, **Lenght** and **Leader**. Leader can be value or **:name**.
Y - read head from tape. Displays type, name, start, lenght and third parameter, and awaits a key. Press **J** to load this block or other key to cancel.

Tracing:
SS+H - call. Monitor asks you for address **Call**, return is possible using ret.
W - set actual address as starting for run with breakpoint.
SS+U - set breakpoint to actual address and jump to address set using W function. Breakpoint takes 3 bytes.
SS+Z - step. Provides instruction at actual address. CALL and RST simulation follows seting (see X function).
T - slow tracing. Provides one instruction, refreshes panel and takes next instruction... Press **BREAK** to stop and **CS+ENTER** to turn display off.
SS+T - fast tracing. Asks for stop address **Last**, provides instructions. Press **BREAK** to stop or **CS+ENTER** to display panel. Stops automatically on stop address.
SS+M - switches EI / DI.
SS+B - exchanges register set (EXX + AX AF,AF').
SS+N - set registers. Question **ld** appears. enter register name, coma or space and new value.
Registers in Monitor are:
One byte: a,b,c,d,e,h,l,hx,lx,hy,ly,i,r
two bytes: af,bc,de,hl,ix,iy,sp
Tacts counter: t
Memory pointers: x,y
Flags: f (to change carry: ld f,c; to change sign: ld f,s; to change zero: ld f,z; to change parity: ld f,p).
1 - set DEFB area. Displays areas 0..4, first and last byte of each. Press number key (**0, 1, 2, 3, 4**) to erase this area or **I** to define new DEFB area. Monitor asks you then for **First** and **Last** byte of area (Last is member of area!).
2 - set DEFW area. Displays areas 0..4, first and last byte of each. Press number key (**0, 1, 2, 3, 4**) to erase this area or **I** to define new DEFW area. Monitor asks you then for **First** and **Last** byte of area (Last is member of area!).
3 - set READ protected area. Displays areas 0..4, first and last byte of each. Press number key (**0, 1, 2, 3, 4**) to erase this area or **I** to define new READ protected area. Monitor asks you then for **First** and **Last** byte of area.
4 - set WRITE protected area. Displays areas 0..4, first and last byte of each. Press number key (**0, 1, 2, 3, 4**) to erase this area or **I** to define new WRITE protected area. Monitor asks you then for **First** and **Last** byte of area.
5 - set RUN protected area. Displays areas 0..4, first and last byte of each. Press number key (**0, 1, 2, 3, 4**) to erase this area or **I** to define new RUN protected area. Monitor asks you then for **First** and **Last** byte of area.

6 - set addresses for direct call (DEF mode, see X function). Displays addresses 0..9. Press number key (**0, 1, 2, 3, 4, 5, 6, 7, 8, 9**) to erase this address or **I** to define new address.
X - change mode of tracing CALL and RST instructions. Modes are three: NON / DEF / ALL. NON means, that no CALL 'll be traced as one instruction with all subroutine. DEF means, that calls to defined addresses 'll be traced as one instruction with subroutine. ALL means, that every CALL performs his subroutine as one instruction.
SS+X - turn all these tests on / off.
SS+W - edit panel.

Panel editor:
Here can you design your own panel.
CS+1 (EDIT) - quit to Monitor.
4 - go to next item.
3 - go to previous item.
5, 6, 7, 8 - move item left, down, up, right.
A to **Z** - set size of item (0..25). It turns registers display on/off, and sets size of dump panels.
SS+D - enables or disables DEC.
SS+H - enables or disables HEX.
SS+B - enables or disables BIN.
SS+C - enables or disables character dump.
SS+T - switches memory dump bytes / words.
SS+S - changes direction of dump - horizontal or vertical.

Item list:
Edit zone (one line)
Dump panel
Disassem panel
Interrupt (EI / DI)
One byte registers (a,b,c,d,e,h,l,i,r,hx,hl,hy,ly,f)
Two bytes registers (af,bc,de,hl,sp,ix,iy)
Tacts counter (T)
Memory dump from pointer (X,Y)
Memory dump from address in register (bc),(de),(hl),(sp),(ix),(iy)