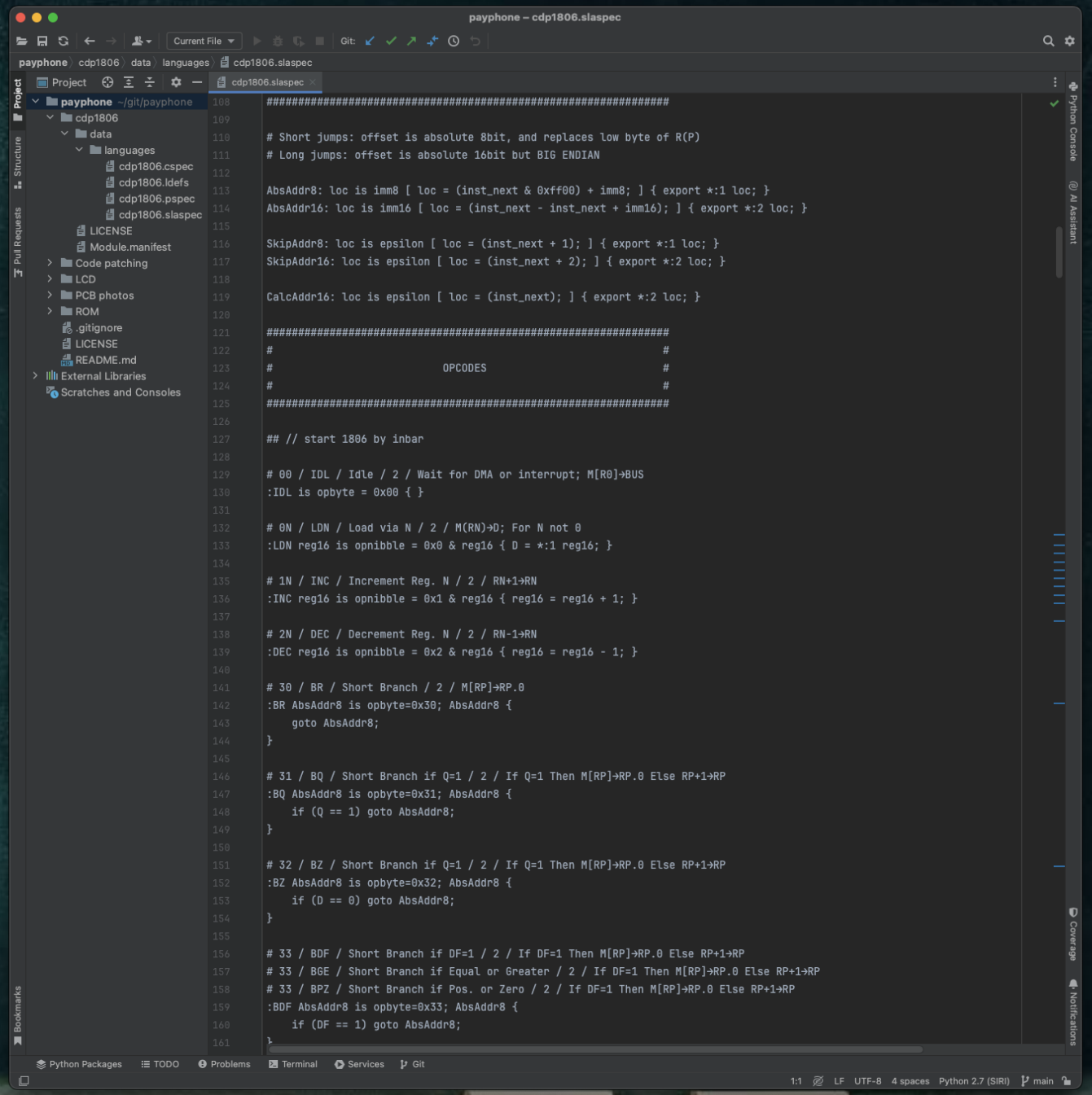
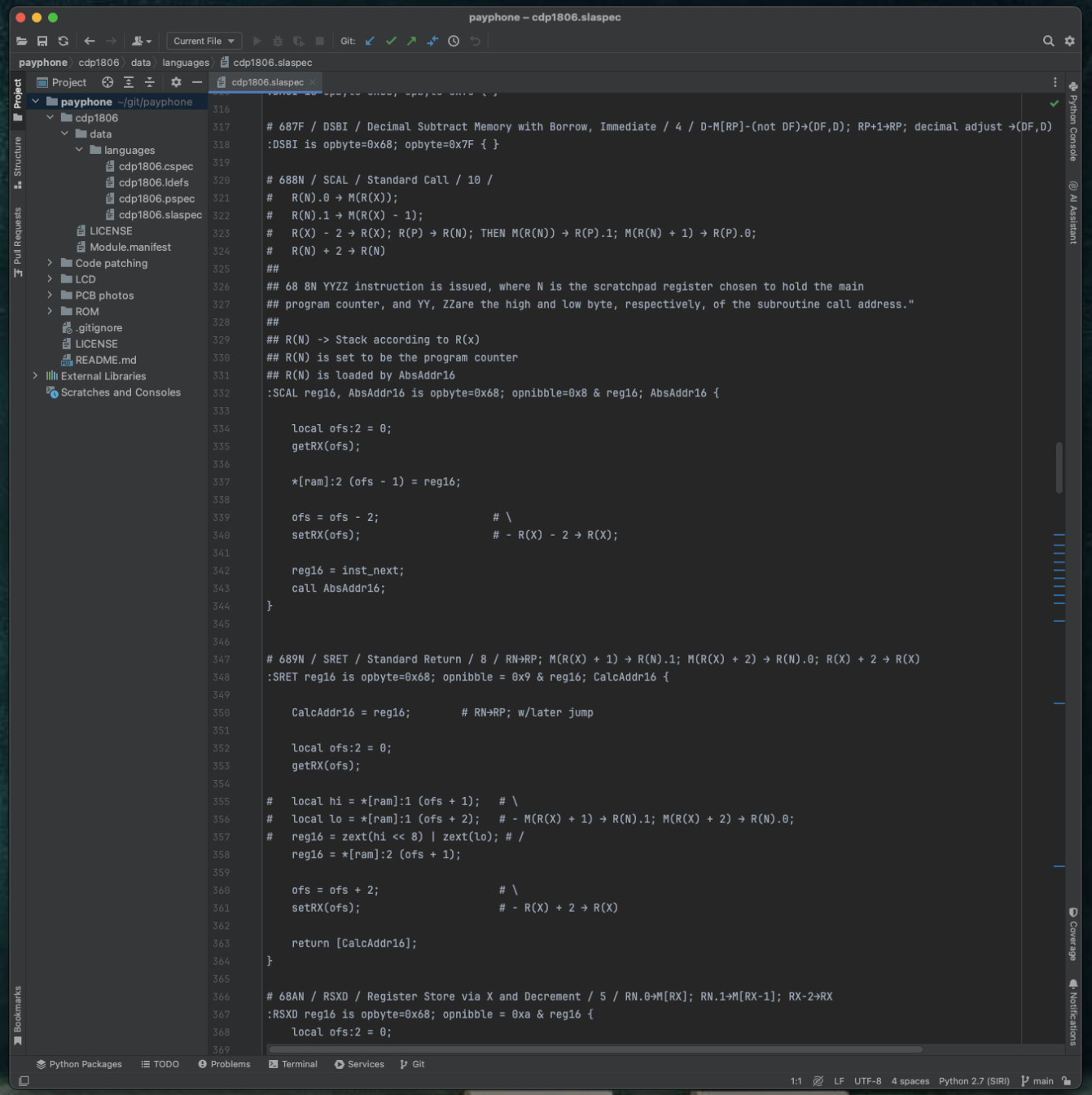
Writing a CPU Module

- Language definitions.
- Architecture definitions.
- Pseudo-instructions and macros.
- Opcodes (simple).



Writing a CPU Module

- Language definitions.
- Architecture definitions.
- Pseudo-instructions and macros.
- Opcodes (simple).
- Opcodes (complex).



Writing a CPU Module

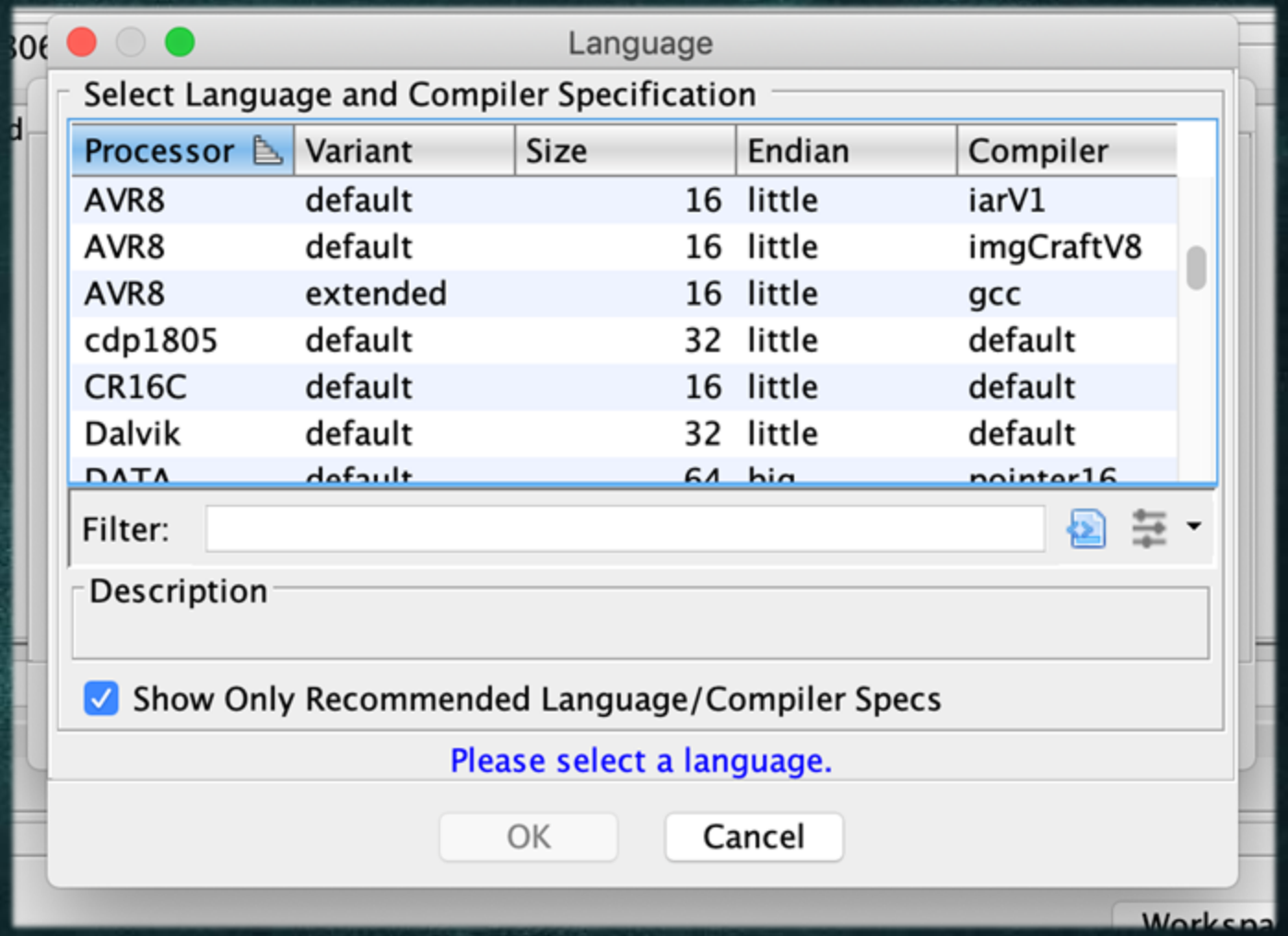
- In mid-November 2019 I flew to Cancun for a confernce.

Writing a CPU Module

- In mid-November 2019 I flew to Cancun for a confernce.
- I was obsessed with getting this done.

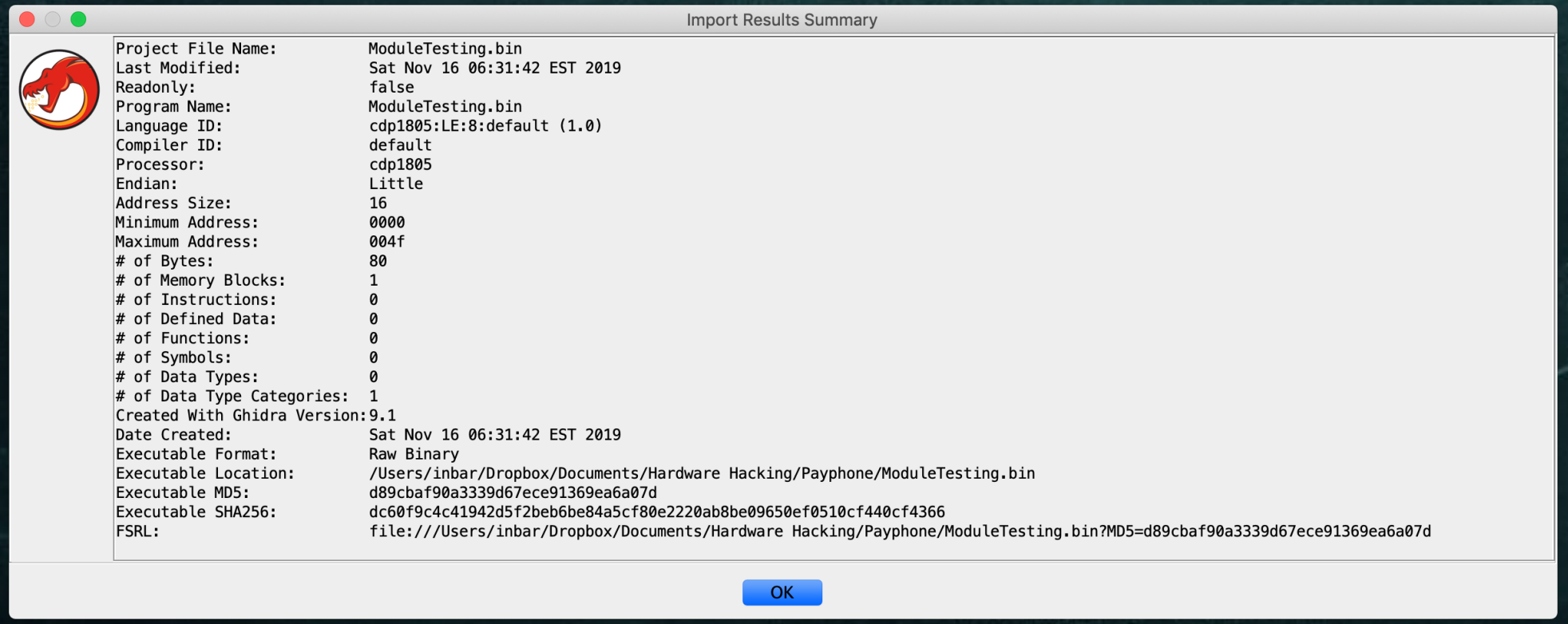
Writing a CPU Module

- First success: cdp1805 in the list.



Writing a CPU Module

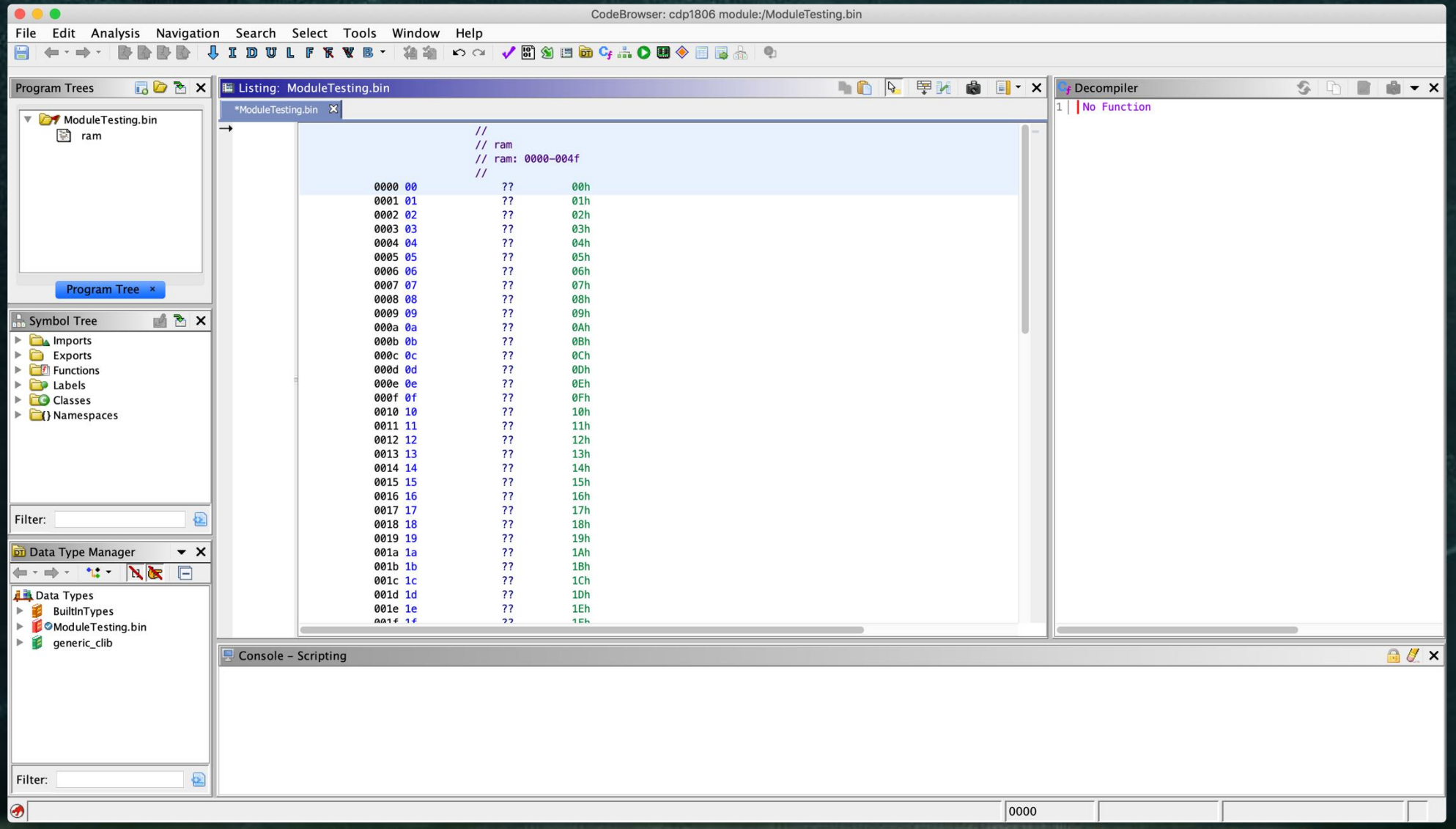
- First success: cdp1805 in the list.
- Successfully loaded binary



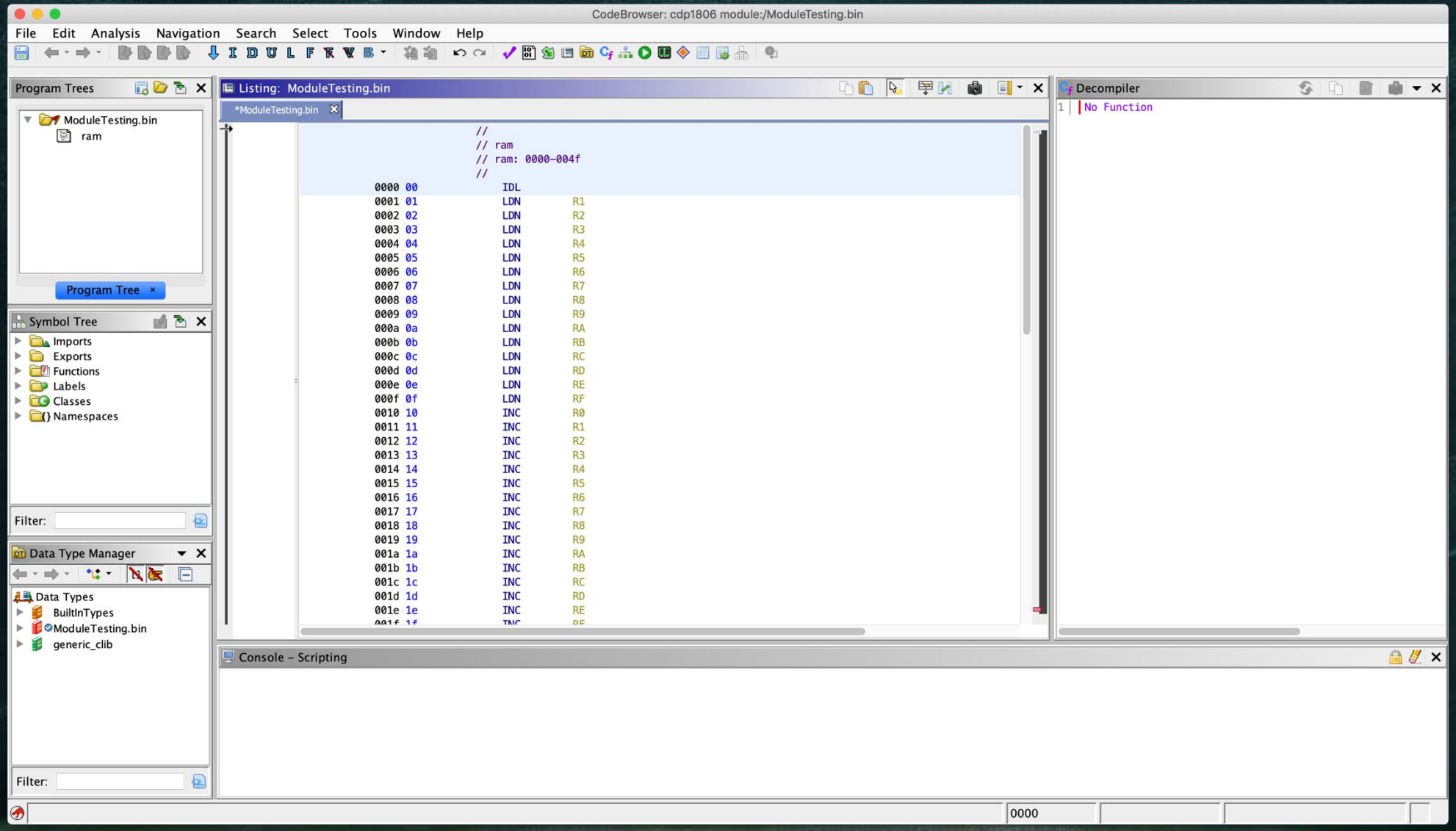
Writing a CPU Module

- First success:
  cdp1805 in the list.
- Successfully loaded binary.
- And the moment of truth…

Writing a CPU Module



Writing a CPU Module



Improving through trial and error

- Disassemble for a while.
- Find an error or a missing feature.
- Modify the CPU definition.
- Reload the binary.

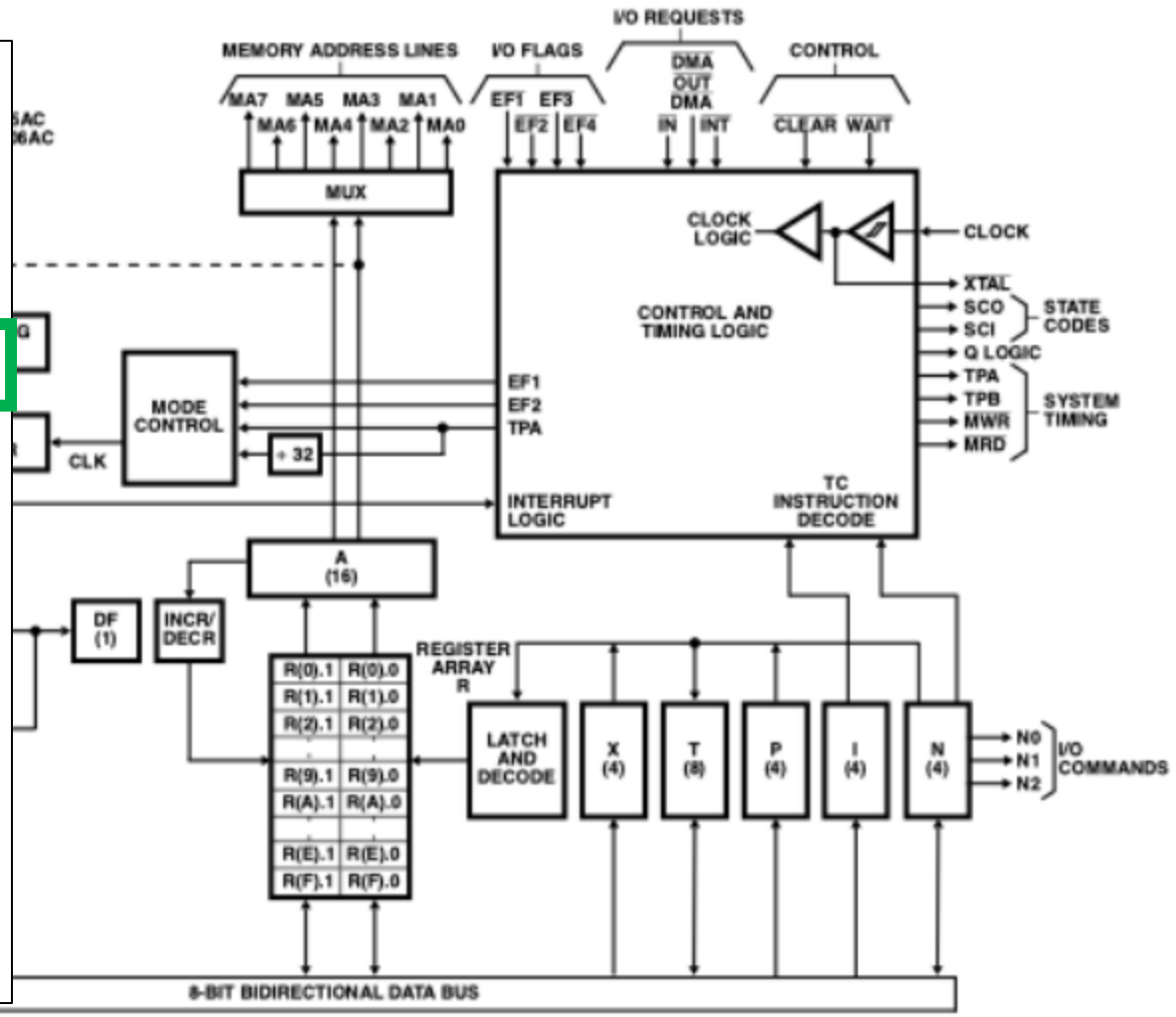
- Update the definition to “cdp1806”.

Reviewing the Peculiarities

ANY General purpose register can be the Program Counter

REGISTER SUMMARY

D	8 Bits	Data Register (Accumulator)
DF	1-Bit	Data Flag (ALU Carry)
B	8 Bits	Auxiliary Holding Register
R	16 Bits	1 of 16 Scratch and Registers
P	4 Bits	Designates which Register is Program Counter
X	4 Bits	Designates which Register is Data Pointer
N	4 Bits	Holds Low-Order Instr. Digit
I	4 Bits	Holds High-Order Instr. Digit
T	8 Bits	Holds old X, P after Interrupt (X is high nibble)
Q	1-Bit	Output Flip-Flop
CNTR	8-Bits	Counter/Timer
CH	8 Bits	Holds Counter Jam Value
MIE	1-Bit	Master Interrupt Enable
CIE	1-Bit	Counter Interrupt Enable
XIE	1-Bit	External Interrupt Enable
CIL	1-Bit	Counter Interrupt Latch



CDP1805AC, CDP1806AC

Examples

- Post-reset code.

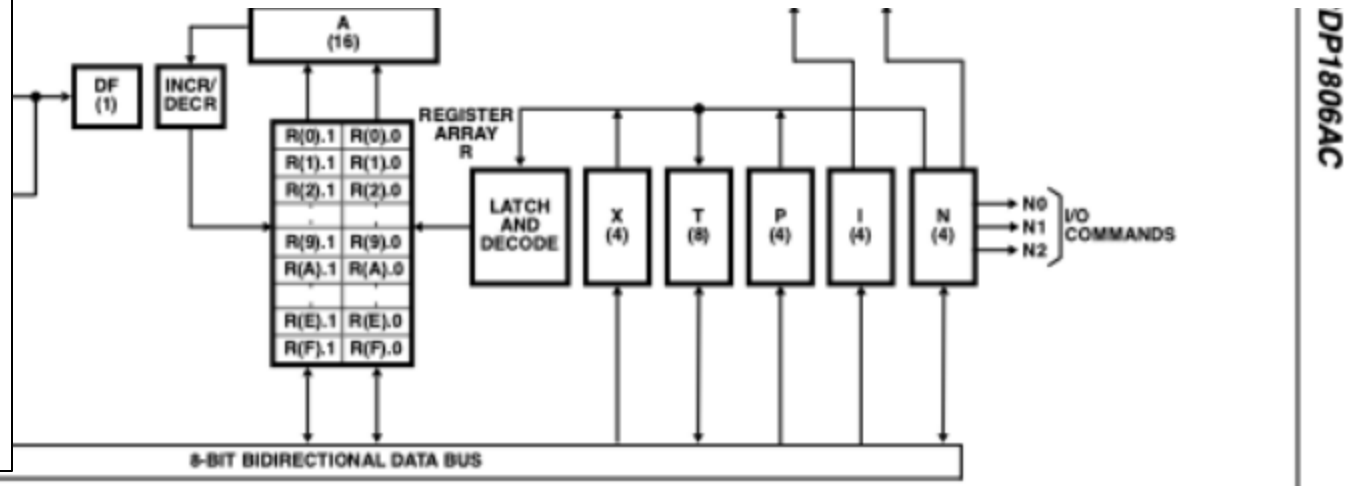
```
        ; After reset, and during a DMA operation, R(0) is used
        ; as the program counter. Performing SEX 0x00 here
        ; means that each OUT opcode
        ; [ M(R(X)) → BUS; R(X) + 1 → R(X) ]
        ; uses the byte after it as the argument.

        start                                           XREF[1]:     0002(j)
6dd5 e0             SEX         0x0
6dd6 7a             REQ

6dd7 65             OUT         0x5                                  0xC0 -> P5
6dd8 c0             db          C0h
6dd9 c4             NOP
6dda c4             NOP

6ddb 62             OUT         0x2                                  0x00 -> P2
6ddc 00             db          0h

6ddd 65             OUT         0x5                                  0x84 -> P5
6dde 84             db          84h
6ddf c4             NOP
6de0 c4             NOP

6de1 62             OUT         0x2                                  0x00 -> P2
6de2 00             db          0h

6de3 65             OUT         0x5                                  0x01 -> P5
6de4 01             db          1h
6de5 c4             NOP
6de6 c4             NOP

6de7 61             OUT         0x1                                  0x00 -> P1
6de8 00             db          0h

6de9 65             OUT         0x5                                  0x04 -> P5
6dea 04             db          4h
6deb c4             NOP
6dec c4             NOP

6ded 63             OUT         0x3                                  0x00 -> P3
6dee 00             db          0h

6def 65             OUT         0x5                                  0x10 -> P5
6df0 10             db          10h
6df1 c4             NOP
6df2 c4             NOP
```

REGISTER SUMMARY

D	8 Bits	Data Register (Accumulator)
DF	1-Bit	Data Flag (ALU Carry)
B	8 Bits	Auxiliary Holding Register
R	16 Bits	1 of 16 Scratch and Registers
P	4 Bits	Designates which Register is Program Counter
X	4 Bits	Designates which Register is Data Pointer
N	4 Bits	Holds Low-Order Instr. Digit
I	4 Bits	Holds High-Order Instr. Digit
T	8 Bits	Holds old X, P after Interrupt (X is high nibble)
Q	1-Bit	Output Flip-Flop
CNTR	8-Bits	Counter/Timer
CH	8 Bits	Holds Counter Jam Value
MIE	1-Bit	Master Interrupt Enable
CIE	1-Bit	Counter Interrupt Enable
XIE	1-Bit	External Interrupt Enable
CIL	1-Bit	Counter Interrupt Latch

CALL AND RETURN				
STANDARD CALL	10	SCAL	688N (Note 10)	R(N).0 → M(R(X)); R(N).1 → M(R(X) - 1); R(X) - 2 → R(X); R(P) → R(N); THEN M(R(N)) → R(P).1; M(R(N) + 1) → R(P).0; R(N) + 2 → R(N)
STANDARD RETURN	8	SRET	689N (Note 10)	R(N) → R(P); M(R(X) + 1) → R(N).1; M(R(X) + 2) → R(N).0; R(X) + 2 → R(X)

Immediate arguments to functions are passed *via the bytes after the call*



Examples

- Inline args.

```
4d73 e2             SEX       0x2
4d74 68 83 74 6d    SCAL      R3,EraseBlock                  Input:
                                                              - word -> Block start
                                                              - word -> Block end (inclusive)
4d78 80 7c          dw        DAT_807c
4d7a 81 85          dw        DAT_8185

4d7c e2             SEX       0x2
4d7d 68 83 74 6d    SCAL      R3,EraseBlock                  Input:
                                                              - word -> Block start
                                                              - word -> Block end (inclusive)
4d81 80 5c          dw        DAT_805c
4d83 80 67          dw        DAT_8067

4d85 e2             SEX       0x2
4d86 68 83 74 6d    SCAL      R3,EraseBlock                  Input:
                                                              - word -> Block start
                                                              - word -> Block end (inclusive)
4d8a 80 00          dw        DAT_8000
4d8c 80 19          dw        DAT_8019
```

Examples

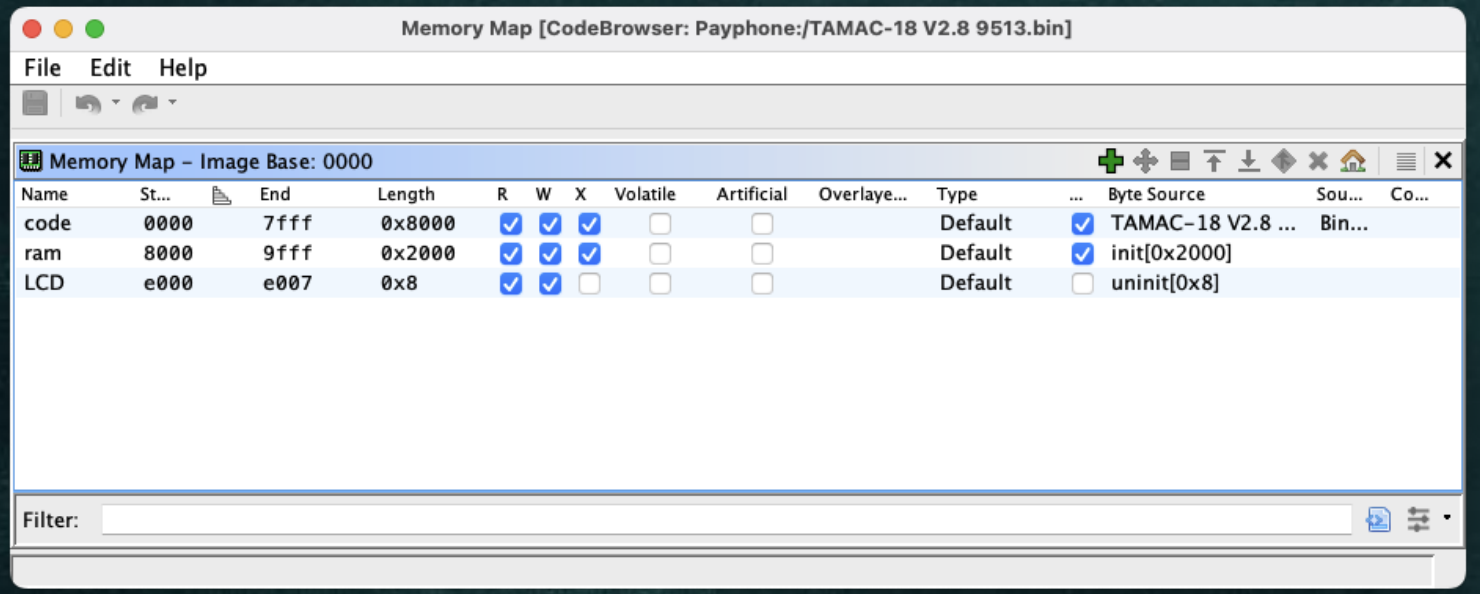
- Inline args.

```
4d73 e2          SEX     0x2
4d74 68 83 74 6d SCAL    R3,EraseBlock


4d78 80 7c       dw      DAT_807c
4d7a 81 85       dw      DAT_8185

4d7c e2          SEX     0x2
4d7d 68 83 74 6d SCAL    R3,EraseBlock


4d81 80 5c       dw      DAT_805c
4d83 80 67       dw      DAT_8067

4d85 e2          SEX     0x2
4d86 68 83 74 6d SCAL    R3,EraseBlock


4d8a 80 00       dw      DAT_8000
4d8c 80 19       dw      DAT_8019
```

```
                *************************************************************
                *                          FUNCTION                         *
                *************************************************************
                undefined EraseBlock()                                  Input:
                                                                         - word -> Block start
                                                                         - word -> Block end (inclusive)
undefined       <UNASSIGNED>   <RETURN>
                EraseBlock                                  XREF[18]:   25cc(R), proc_2f04:2f05(R),
                                                                        FUN_2f26:2f3b(R),
                                                                        FUN_2f26:2f44(R), 3395(R),
                                                                        3407(R), FUN_4ca6:4ce3(R),
                                                                        4d74(R), 4d7d(R), 4d86(R),
                                                                        4e36(R), proc_57b6:57c9(R),
                                                                        5fc1(R), 6e2a(R), 71ac(R),
                                                                        FUN_74a7:74cc(R),
                                                                        FUN_74a7:74d5(R),
                                                                        proc_7640:765f(R)
        746d f8 01      LDI     0x1                                             Input:
                                                                                 - word -> Block start
                                                                                 - word -> Block end (inclusive)
        746f cc         LSIE    LAB_7472
        7470 f8 00      LDI     0x0

            LAB_7472                                        XREF[1]:    746f(j)
        7472 73         STXD
        7473 c2 74 79   LBZ     LAB_7479

        7476 e4         SEX     0x4
        7477 71         DIS
        7478 24         DEC     R4

            LAB_7479                                        XREF[1]:    7473(j)
        7479 e3         SEX     0x3
        747a 68 6d      RLXA    RD                                              arg1
        747c 68 6c      RLXA    RC                                              arg2
        747e e2         SEX     0x2
        747f 68 83 74 98 SCAL   R3,CalcBlockSize                                RD = (RC - RD) + 1
                                                                                Input (SEX 02):
                                                                                 - R2 -> scratchpad
                                                                                 - RC -> Block last byte
                                                                                 - RD -> Block first byte


                ; Fill the block with 00
```

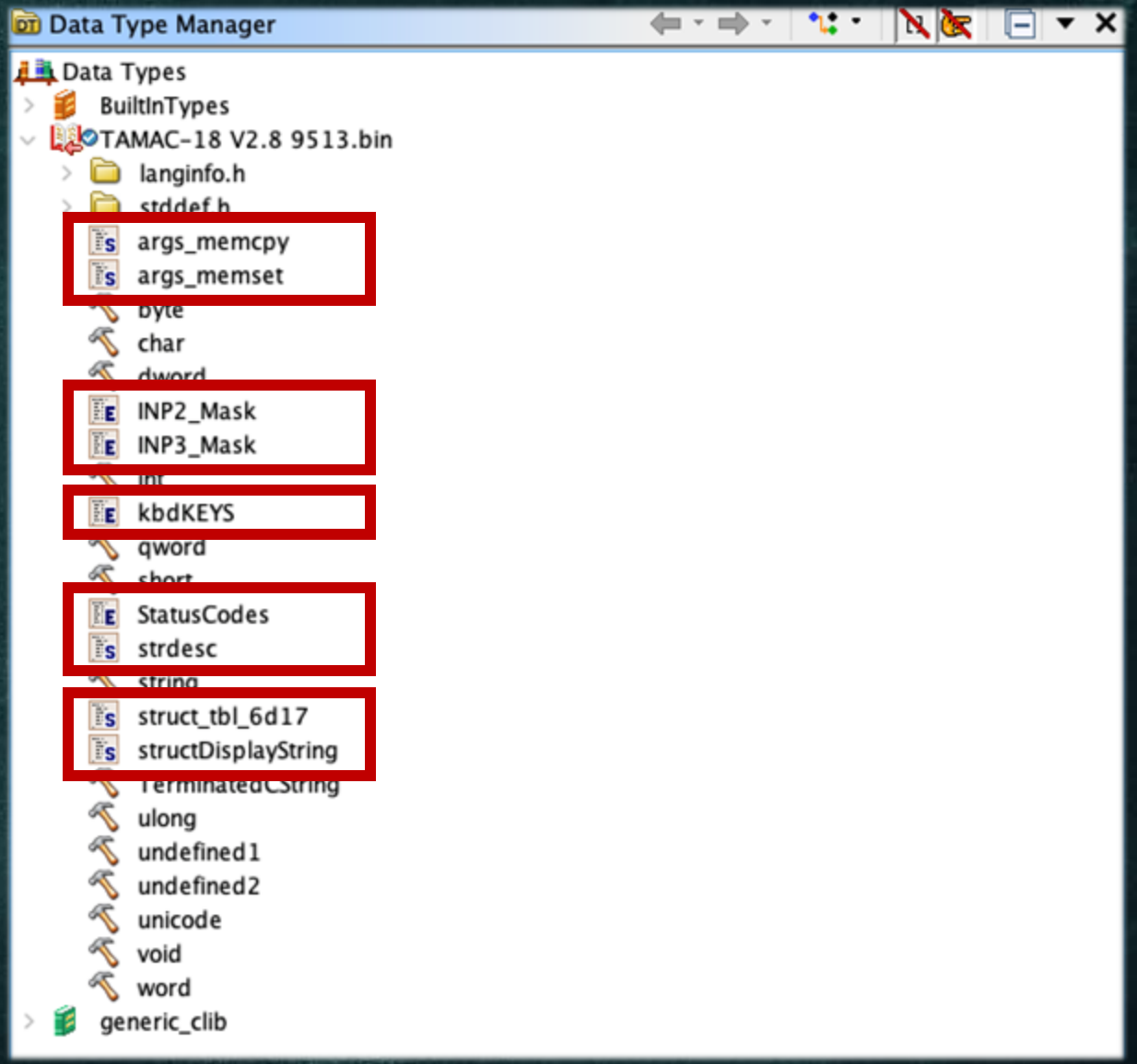
Other Examples

- Memory Map.



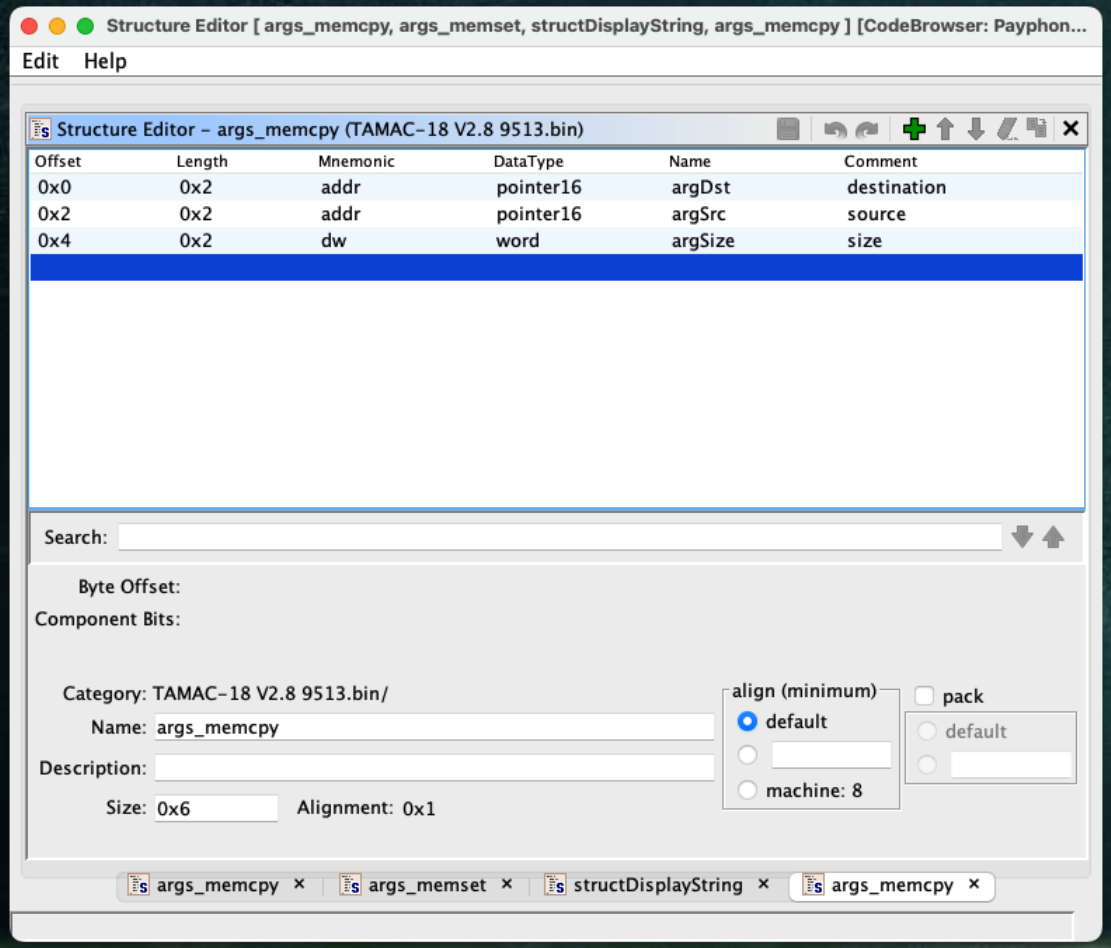
Other Examples

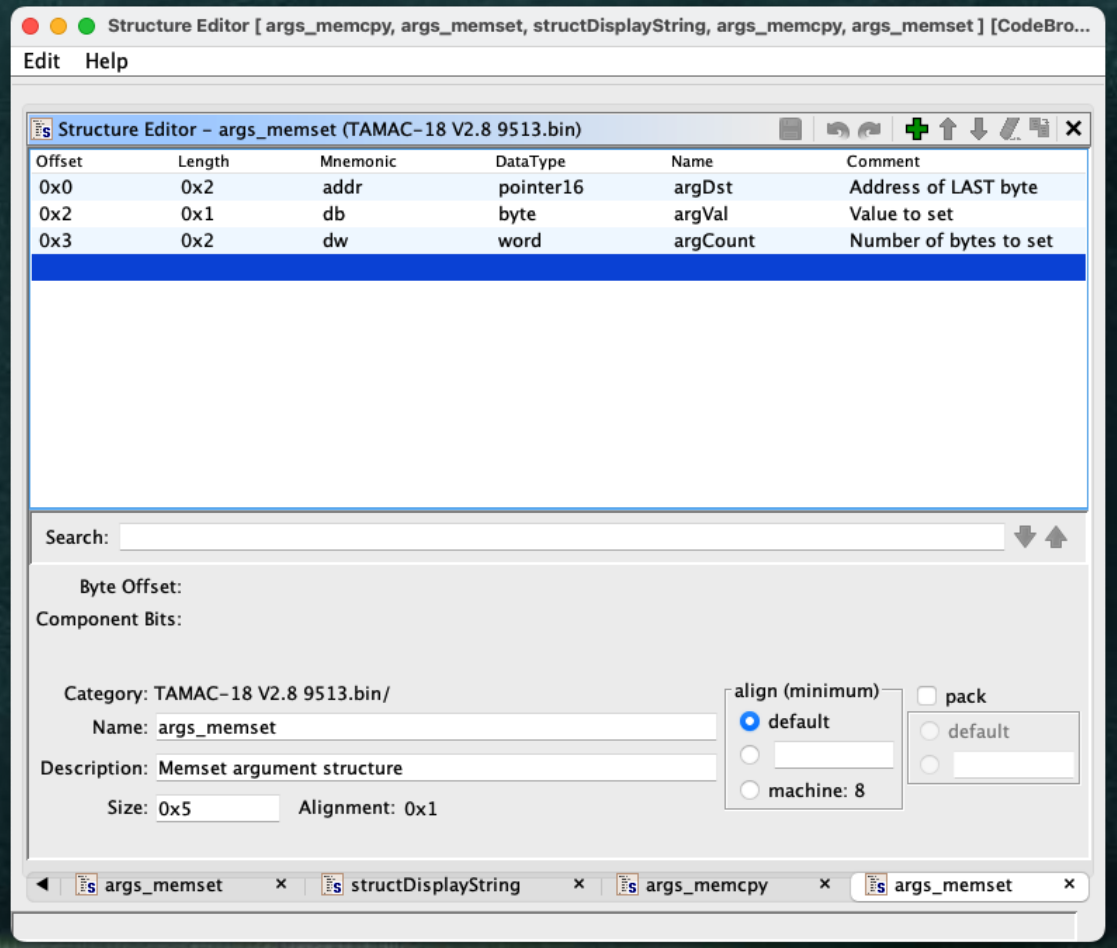
- Data types.



Other Examples

- Data types.
  - Memory operations





Other Examples

- Basic functions.

```
                        *************************************************************
                        *                          FUNCTION                         *
                        *************************************************************
                        undefined isDigit?()
undefined               <UNASSIGNED>   <RETURN>
                        isDigit?                                XREF[1]:     3e8a(R)
       385a ff 30           SMI         0x30
       385c cb 38 62        LBNF        LAB_3862

       385f ff 0a           SMI         0xa
       3861 c7              LSNF        LAB_3864

                        LAB_3862                                XREF[1]:     385c(j)
       3862 f8 00           LDI         0x0

                        LAB_3864                                XREF[1]:     3861(j)
       3864 68 93           SRET        R3
```

```
                        *************************************************************
                        *                          FUNCTION                         *
                        *************************************************************
                        undefined popcnt()                              Input:
                                                                         - D - Byte to perform popcnt() on
                                                                        Output:
                                                                         - D - popcnt(D)
undefined               <UNASSIGNED>   <RETURN>
                        popcnt                          XREF[3]:     33cc(R), FUN_5426:5457(R),
                                                                     FUN_5bfd:5cf4(R)
       3886 68 cd 00 08     RLDI        RD,0x8                              Input:
                                                                             - D - Byte to perform popcnt() on
                                                                            Output:
                                                                             - D - popcnt(D)
       388a 68 cc 00 00     RLDI        RC,0x0

                        LAB_388e                        XREF[1]:     3892(j)
       388e f6              SHR
       388f c7              LSNF        LAB_3892
       3890 1c              INC         RC
       3891 c4              NOP

                        LAB_3892                        XREF[1]:     388f(j)
       3892 68 2d 38 8e     DBNZ        RD,LAB_388e

       3896 8c              GLO         RC
       3897 e2              SEX         0x2
       3898 68 93           SRET        R3
```

```
                        *************************************************************
                        *                          FUNCTION                         *
                        *************************************************************
                        undefined isInRange()                                   Input (SEX R2):
                                                                                 - byte - Low limit
                                                                                 - byte - High limit
                                                                                 - D - Value to test
                                                                                 - R2 - storage for D
undefined               <UNASSIGNED>   <RETURN>
                        isInRange                                XREF[6]:     FUN_5426:542b(R),
                                                                              FUN_5426:543f(R),
                                                                              FUN_5426:546d(R),
                                                                              FUN_5426:5481(R),
                                                                              FUN_5426:5491(R),
                                                                              FUN_54b5:54d8(R)
       558d 52              STR         R2                                   R2 = D

       558e 43              LDA         R3                                   D = arg1
       558f f5              SD                                               M(R(X)) - D → DF, D
       5590 c7              LSNF        LAB_5593                             jmp if M(RX) >= D

       5591 43              LDA         R3                                   arg2
       5592 f7              SM                                               D = M(RX) → DF, D
                                                                             D -= M(R2)

                        LAB_5593                                XREF[1]:     5590(j)
       5593 c3 55 99        LBDF        LAB_5599                             jmp if borrow

       5596 f8 00           LDI         0x0
       5598 c8              LSKP        LAB_559b

                        LAB_5599                                XREF[1]:     5593(j)
       5599 f8 01           LDI         0x1

                        LAB_559b                                XREF[1]:     5598(j)
       559b c4              NOP
       559c c4              NOP
       559d 68 93           SRET        R3
```

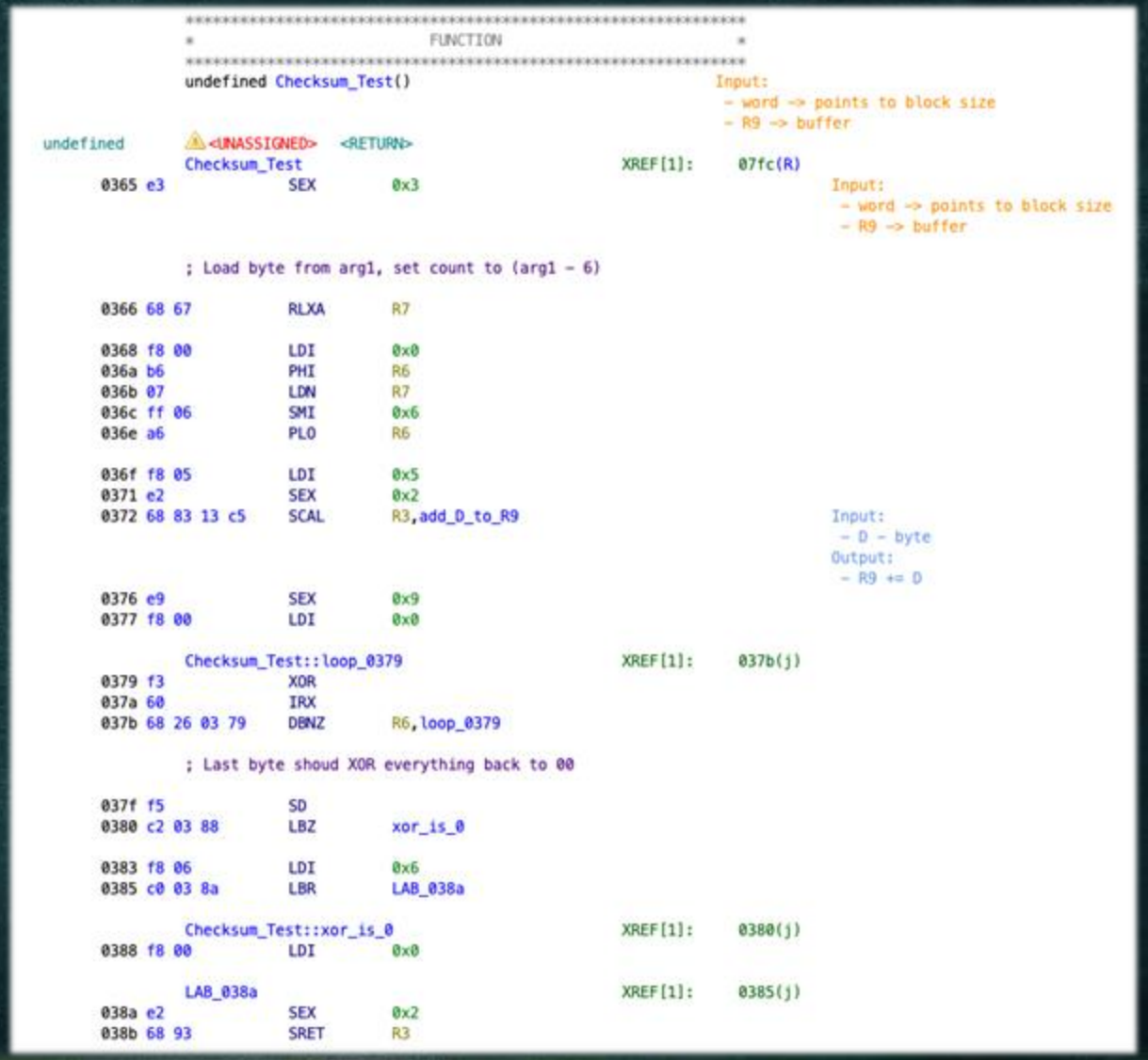
Other Examples

- Interesting functions.

```
                ****************************************************************
                *                          FUNCTION                            *
                ****************************************************************
                undefined Add_Words()                                  Input:
                                                                        - word -> ptr to word1.0
                                                                        - word -> ptr to word2.0
                                                                       Output:
                                                                        - word2 += word1
undefined       ⚠<UNASSIGNED>   <RETURN>
                Add_Words
       38ab e3          SEX         0x3                                    Input:
                                                                            - word -> ptr to word1.0
                                                                            - word -> ptr to word2.0
                                                                           Output:
                                                                            - word2 += word1
       38ac 68 6c       RLXA        RC                                     arg1
       38ae 68 6d       RLXA        RD                                     arg2
       38b0 ed          SEX         0xd

       38b1 0c          LDN         RC                                     D = *RC
       38b2 2c          DEC         RC                                     RC--
       38b3 f4          ADD                                                M(RD) + D → DF, D
       38b4 73          STXD                                               *RD-- = D

       38b5 0c          LDN         RC                                     D = *RC
       38b6 74          ADC
       38b7 73          STXD                                               *RD-- = D

       38b8 e2          SEX         0x2
       38b9 68 93       SRET        R3
```

Other Examples

- Interesting functions.



```
                        ****************************************************************
                        *                          FUNCTION                            *
                        ****************************************************************
                        undefined Checksum_Test()                                       Input:
                                                                                         - word -> points to block size
                                                                                         - R9 -> buffer
undefined               <UNASSIGNED>  <RETURN>
                        Checksum_Test                                   XREF[1]:     07fc(R)
      0365 e3                   SEX       0x3                                             Input:
                                                                                          - word -> points to block size
                                                                                          - R9 -> buffer

                        ; Load byte from arg1, set count to (arg1 - 6)

      0366 68 67                RLXA      R7

      0368 f8 00                LDI       0x0
      036a b6                   PHI       R6
      036b 07                   LDN       R7
      036c ff 06                SMI       0x6
      036e a6                   PLO       R6

      036f f8 05                LDI       0x5
      0371 e2                   SEX       0x2
      0372 68 83 13 c5          SCAL      R3,add_D_to_R9                                  Input:
                                                                                          - D - byte
                                                                                        Output:
                                                                                          - R9 += D

      0376 e9                   SEX       0x9
      0377 f8 00                LDI       0x0

                        Checksum_Test::loop_0379                        XREF[1]:     037b(j)
      0379 f3                   XOR
      037a 60                   IRX
      037b 68 26 03 79          DBNZ      R6,loop_0379

                        ; Last byte shoud XOR everything back to 00

      037f f5                   SD
      0380 c2 03 88             LBZ       xor_is_0

      0383 f8 06                LDI       0x6
      0385 c0 03 8a             LBR       LAB_038a

                        Checksum_Test::xor_is_0                         XREF[1]:     0380(j)
      0388 f8 00                LDI       0x0

                        LAB_038a                                        XREF[1]:     0385(j)
      038a e2                   SEX       0x2
      038b 68 93                SRET      R3
```

Other Examples

- Interesting functions.

REMEMBER
this one

```
                        ****************************************************************
                        *                          FUNCTION                            *
                        ****************************************************************
                        undefined byte_to_hex_ascii()                          Input:
                                                                                - D -> byte
                                                                                - R2 -> dest to receive separate nibbles
                                                                                - RA -> dest to receive ASCII HEX
undefined               <UNASSIGNED>   <RETURN>
                        byte_to_hex_ascii                          XREF[2]:     3c59(R), 3c8f(R)
      3d18 73           STXD                                                    Save D at M[RX]
      3d19 fa f0        ANI       0xf0
      3d1b f6           SHR
      3d1c f6           SHR
      3d1d f6           SHR
      3d1e f6           SHR
      3d1f 52           STR       R2                                            Save high nibble in M[R2]
      3d20 ff 09        SMI       0x9
      3d22 c2 3d 2c     LBZ       LAB_3d2c
      3d25 cb 3d 2c     LBNF      LAB_3d2c
      3d28 52           STR       R2
      3d29 f8 40        LDI       0x40
      3d2b c8           LSKP      LAB_3d2e

                        LAB_3d2c                                   XREF[2]:     3d22(j), 3d25(j)
      3d2c f8 30        LDI       0x30

                        LAB_3d2e                                   XREF[1]:     3d2b(j)
      3d2e f4           ADD
      3d2f 5a           STR       RA
      3d30 1a           INC       RA
      3d31 12           INC       R2
      3d32 02           LDN       R2
      3d33 fa 0f        ANI       0xf
      3d35 52           STR       R2
      3d36 ff 09        SMI       0x9
      3d38 c2 3d 42     LBZ       LAB_3d42
      3d3b cb 3d 42     LBNF      LAB_3d42
      3d3e 52           STR       R2
      3d3f f8 40        LDI       0x40
      3d41 c8           LSKP      LAB_3d44

                        LAB_3d42                                   XREF[2]:     3d38(j), 3d3b(j)
      3d42 f8 30        LDI       0x30

                        LAB_3d44                                   XREF[1]:     3d41(j)
      3d44 f4           ADD
      3d45 5a           STR       RA
      3d46 1a           INC       RA
      3d47 68 93        SRET      R3
```

Decoding Hebrew

- Language-specific encoding in the 80's and 90's was defined through "Code Pages".

Code page

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

In computing, a **code page** is a character encoding and as such it is a specific association of a set of printable characters and control characters with unique numbers. Typically each number represents the binary value in a single byte. (In some contexts these terms are used more precisely; see Character encoding § Terminology.)

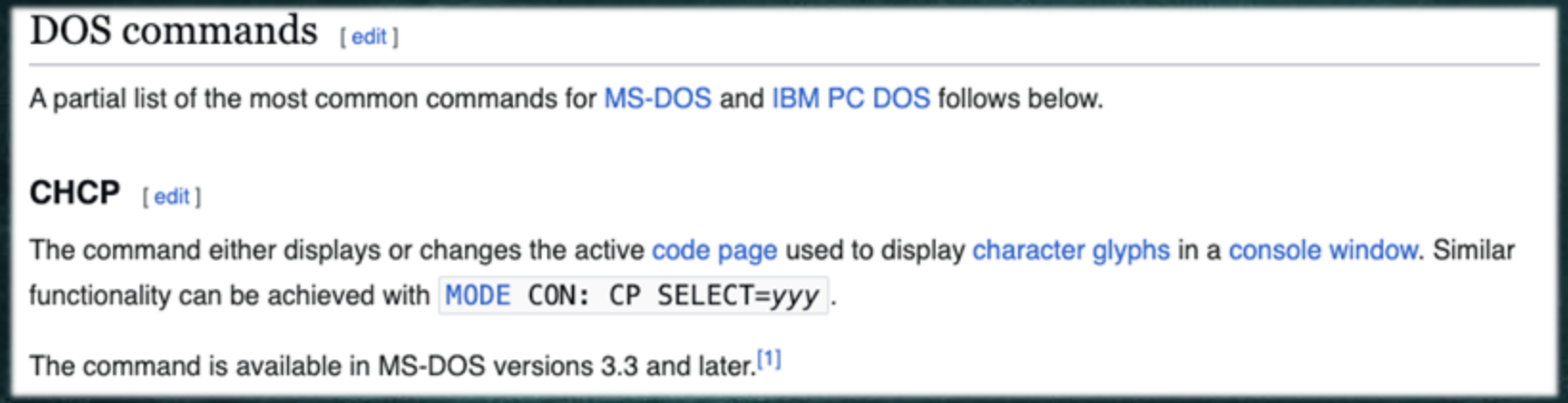
The term "code page" originated from IBM's EBCDIC-based mainframe systems,[1] but Microsoft, SAP,[2] and Oracle Corporation[3] are among the vendors that use this term. The majority of vendors identify their own character sets by a name. In the case when there is a plethora of character sets (like in IBM), identifying character sets through a number is a convenient way to distinguish them. Originally, the code page numbers referred to the *page* numbers in the IBM standard character set manual,[4][5][6] a condition which has not held for a long time. Vendors that use a code page system allocate their own code page number to a character encoding, even if it is better known by another name; for example, UTF-8 has been assigned page numbers 1208 at IBM, 65001 at Microsoft, and 4110 at SAP.

Hewlett-Packard uses a similar concept in its HP-UX operating system and its Printer Command Language[7] (PCL) protocol for printers (either for HP printers or not). The terminology, however, is different: What others call a *character set*, HP calls a *symbol set*, and what IBM or Microsoft call a *code page*, HP calls a *symbol set code*. HP developed a series of symbol sets,[8][9] each with an associated symbol set code, to encode both its own character sets and other vendors' character sets.

The multitude of character sets leads many vendors to recommend Unicode.

Decoding Hebrew

- Language-specific encoding in the 80's and 90's was defined through "Code Pages".
- If you're old enough, you remember the DOS command "CHCP".



DOS commands [edit]

A partial list of the most common commands for MS-DOS and IBM PC DOS follows below.

CHCP [edit]

The command either displays or changes the active code page used to display character glyphs in a console window. Similar functionality can be achieved with `MODE CON: CP SELECT=yyy`.

The command is available in MS-DOS versions 3.3 and later.[1]