People 6502 Forth Assembler #

(Work in progress, pending translation)

Source: 6502 assembler

See also: 6502 assembler in Forth

TabbedSection Tab-english tab content 1 /% Tab-German

Table of Contents

The 6502 assembler #

In the following the concepts of the 6502 Forth Assembler for the people are represented. It is given a complete glossary, as the assembler mnemonics of all machine-language programmers may be unfamiliar. A detailed description of the operation is found in the FORTH DIMENSIONS, Vol III, p. 5 143ff. The following is a brief summary given and presented changes to the original.

The operation of the interpreter and address of the routine NEXT is presented in Chapter 2 of the People's Forth Guide. The 6502 assembler allows structured programming. The structural elements are constructed analogous to the control structures of the Forth but bear different names in order to reduce the risk of confusion and increase clarity.

For example:

```
cc? [<ausdruck1>] [<ausdruck2>]?
```

cc stands for "condition code". <ausdruck1> is executed if cc is true, otherwise <ausdruck2>. The part

```
] [<ausdruck2>
```

can also be omitted. The analogue in Forth is IF ... ELSE THEN

Please note, that must stand before ?[always (!) A condition code. There will also be no checks for proper nesting of control structures.

Other control structures are: ((([<ausdruck1> Cc? [<ausdruck2>]]? [<ausdruck1> Cc?] [<ausdruck1>]])))

The analogous expressions in Forth are: (((<ausdruck1> BEGIN WHILE REPEAT <ausdruck2> BEGIN UNTIL <ausdruckl> BEGIN REPEAT <ausdruck1>))) Here, too, must not be left out at the assembler cc words. Moreover, exactly one ?[[between [[and]]? is permitted. Also note the difference between]] and]]?!

As a condition code will be accepted: (((0 = 0 <> 0 <0> = CS CC VS VC)))

Find the processor flags ZNC and V are assigned. The first example will be executed <ausdruck1> if cc is replaced by 0 =, and the Z flag is set. Each condition codes can be extended by one following NOT, for example: (((0 = NOT? 0 lda #?)))

In addition to all other opcodes with their addressing modes, there are the jumps BCC, BCS, etc. You are only addressing the "Absolute" is allowed, ie on the stack is the address of the jump target. If this address exist-ing outside the area, so the error message is displayed "out of range".

For the other opcodes are permissible according to the command, the following addressing modes: (((. A

. X , Y X)) Y)))) Addressing the "absolute" is used when no other is specified. Used with an unauthorized one mnemonic addressing, so the error message "invalid" output.

Examples of using the 6502 assembler (for explaining the conventional notation is added):

(((. A rol -> rol a Ldy # 1 -> ldy # 1 data, X sta -> static data, x $ 6 x) adc -> adc ($ 6, x) vector) y lda -> lda (vector), y vector) jmp -> jmp (vector))))

Additionally, the system contains several macros that can address all the only "absolute":

Winc #

Increments a 16-bit pointer by 1 wdec decremented analogously.

2inc #

Increments a 16-bit pointer by 2 2dec decremented analogously.

C: #

Turns off the assembler and Forth compiler makes it possible to machine code in Forth-versa. A counterpart is not present.

An example of the use of (((C:))) is: (((... 0 <? No InterWiki reference defined in properties for Wiki called "C"!? ...)))

Is anything less than zero, then the "error" is printed and aborted the execution of the word, otherwise it goes in the code.

> LABEL and LABEL #

Finally there are the words (((> LABEL))) and (((LABEL))). (((> LABEL))) a label created in the heap, and it takes the value of the label from the stack. (((LABEL))) generates a label with a value of countries HERE. Example: (((Label loop index bne loop)))

NEXT #

A code word must always ultimately lead to (()))) NEXT JMP to the address interpreter is working on. Glossary The following constants are specified, you can jump on and bring the values on the stack or removed from it. Particularly important is the routine SETUP. It copies the number of values that is specified in the accumulator into the memory location N.

Access to the stack #

For access to the stack, as far as is possible, we recommend the use of the words (((SETUP))) and (((PUSH ...}}}. This is however often not enough. In this case, the values on the stack to access the following: (((SP x) lda \ The lower byte of the first value SP) y lda \ The upper byte of the first value))) and by setting the Y-register and the second, third, etc. values. Please note that is demanded in NEXT in the X-register the content of $ 00 and the Y-register $ 01 has the content. This was exploited in the above example.

Examples of assembly code in Forth, which we regard as good, among others, the words (((FILL))) and ((()))-TRAILING (popularly Forth kernel). Do you want to program assembly language, so you should look at these words and a few others.

ROM access to Commodore machines #

In assembler programming must beachtetwerden that people Forth off the ROM (for Commodore machines). Therefore, read accesses to the ROM warden somewhat differently. Example of the C16: (((ffd2 jsr \ jumps to a RAM routine. ff3e ff3f sta sta ffd2 jsr \ jumps to the ROM routine.))) It only works when it is in the lower RAM area (<$ 8000). Otherwise follow undefined reactions.

When a C64 bank switching only for read access in the BASIC ROM is required. It should be noted also that a bank switching with (((SEI))) has to be prepared, as was otherwise the periodic Tastaturinterrupt lead to a crash.

The C16 (((no))) SEI is necessary because in the RAM of the vector $ FFFE to interrupt their own shows (she Needed about a thousand of the computation). For the same reason a BRK instruction leads while still in the monitor, but with the wrong Registerdump as the monitor finds dam on the data of the interrupt stack instead of registers.

Glossary #

Pusha (- addr) #

A constant containing the address of a machine code sequence that sets the sign extended contents of the battery on the data stack and then jumps to NEXT. Is used as the last branch instruction in code-words.

Push0A (- addr) #

A constant which Maschinencade proposal contains the address of a sequence which bears the contents of the battery on the low byte of the data stack. The high byte is always set to 0. Then to jump to NEXT. Is used as the last branch instruction in code-words.

Push (- addr) #

A constant containing the address of a machine code sequence that sets the contents of the battery as a high-byte on the data stack. The low byte is fetched from the processor stack, and there must first have been stored. Then to jump to NEXT. Is used as the last branch instruction in code-words.

Next (- addr) #

A constant that specifies the address of NEXT on the data stack. Is used as the last command in code words.

xyNext (- addr) #

As NEXT, but before the X register are loaded with 0 and the Y-register by 1. The system is expected to rule on appeal of NEXT those values in the registers. Otherwise, it responds with a crash.

Puta (- addr) #

A constant containing the address of a machine code sequence which bears the battery as a low-bytes on the data stack. The high byte is not changed, just as, in contrast to PUSH created no space on the data stack. Is then run through NEXT. Is used as the last command in code words.

Pop (- addr) #

A constant containing the address of a machine code sequence which removes the top element from the data stack. Is then used to NEXT gesprungen.Wird as the last branch instruction in code-words.

PopTwo (- addr) #

Einc Kanstante, which contains the adressee Maschinencade a sequence which removes the top two elements from the data stack. Then to jump to NEXT. Is used as the last branch instruction in code-words.

RP (- addr) #

A constant containing the address of the return stack pointer.

UP (- addr) #

A constant containing the address of the user pointers, so the offset to ORIGIN.

SP (- addr) #

A constant containing the address of the data stack pointer.

IP (- addr) #

A constant containing the address of the instruction pointer of the Forth machine. This points to the next work off each word.

W (- addr) #

A constant containing the address of the Forth word pointer machine. This points to the currently processed each word.

N (- addr) #

A constant which contains the address of a storage area in the Zeropage that the user is available.

setup (- addr) #

A constant containing the address of a machine code sequence, the n Eleaente degrades the data stack, and store at N. The number n must be in the battery when SETUP is jumped as a subroutine. The top element of the data stack is N, the second at N +2, etc. In the end, X-and Y-register is set to 0 or 1.

WCMP (addr1 addr2 -) #

The Macro Assembler assembles a sequence that compares the contents of execution of the word with the contents of the word addr1 addr2 on. Then, the carry flag is high when the contents of addr1 greater than or equal to the content of addr2 is. It will change the battery and the status register flags CZON.

ram (-) #

Commodore

Macro. Turns off when running on a different memory bank. The exact mode of action is machine-dependent.

rom (-) #

Commodore

Macro. Turns off when running on a different memory bank. The exact mode of action is machine-dependent.

sys (addr -) #

Commodore

Macro. Turns off when run on a bank using system routines and leads to a jump from addr. The exact mode of action is machine-dependent.

/% /%