Assembler

2013/9/29
Reference: Chapter7 in Propforth.html
asm_demo.f

How to use assembler

Assembler code

```
fl
build_BootOpt :rasm
  - assembler statement -
;asm a_delay                  <-- a_delay is assembler-word's name
```

1. Copy/paste or Load asm.f inside ~/V5.5/CurrentRelease/Extensions/

2. Copy/paste assembler code to TeraTerm
(When using assembler-word, its assembler-list always must paste on source-code's end)

3. As asm.f print out asm-source as 64Base-code, you paste this to source-code using asm_word.

Caution;

Assembler for PropForth is a little different from PASM.
PAR-register don't use.
Data pass through stack same as forth-word.
When making assembler-word, you should not do "swap,tuck,over".
When out of assembler-word to kernel, data on stack should be one or nothing.
If you want data more than two, you should write asm-code to save in HUB-ram.

Assembler word"a_tick" return loop-ticks.

```
\ Delay 72ticks
__delay
        mov      cnt , # d59    4ticks    using writable register[shadow register for cnt]
                                minumum is 5
                                If less than 5 this rollover at waicnt
        add      cnt , cnt      4ticks
        waitcnt  cnt , # 0      6ticks + 54ticks(72-18)

__delay_ret
ret                            4ticks
```

```
Prop0 Cog6 ok
1 ticks
88 Prop0 Cog6 ok
2 ticks
168 Prop0 Cog6 ok
10 ticks
808 Prop0 Cog6 ok
100 ticks
8008 Prop0 Cog6 ok
1000 ticks
80008 Prop0 Cog6 ok
```

Measure overhead-time it takes to palce data on stack.

Refer 'time1,time2,........time1000'
When 1 or 2 on stack, it takes d496 ticks.
When >3 on stack, it takes d368 ticks.

I have no idea why value is different.

Measure overhead-time on 'a_delay_1'

Assembler word"a_delay1" return delay-count-ticks for forth-kernel.
Since this ticks use forth-kernel, it include overhead-time for calling.
It' strange less than 8 a_delay_1.

```
Prop0 Cog6 ok
delay1_1   delay1_2 delay1_3 delay1_4 delay1_5 delay1_6 delay1_7 delay1_8 delay1_9 delay1_10
352 432 512 592 672 752 832 912 992 1072 Prop0 Cog6 ok
delay1_20 delay1_30 delay1_40 delay1_50 delay1_60 delay1_70 delay1_80 delay1_90 delay1_100
delay1_500 delay1_1000
1872 2672 3472 4272 5072 5872 6672 7472 8272 40272 80272 Prop0 Cog6 ok
```

Each value include 80ticks * n.   (n=1,2,... 1000)
So, overhead-time(It takes time when forth-kernel calling asm_word) is 272ticks.

Measure overhead-time on 'a_delay'

Assembler word"a_delay" return delay-count-ticks for forth-kernel.
Data on stack of "a_delay" is more than 9.

```
Prop0 Cog6 ok
delay_9 delay_10 delay_11 delay_12 delay_13 delay_14 delay_15 delay_16 delay_17 delay_18
384 464 544 624 704 784 864 944 1024 1104 Prop0 Cog6 ok
delay_28 delay_38 delay_48 delay_58 delay_68 delay_78 delay_88 delay_98 delay_108 delay_508
delay_1008
1904 2704 3504 4304 5104 5904 6704 7504 8304 40304 80304 Prop0 Cog6 ok
```

Each value include 80ticks * n.   (n=1,2,... 1000)
So, overhead-time(It takes time when forth-kernel calling asm_word) is 304ticks.

272ticks at "a_delay1",   304ticks at "a_delay"
304 – 272 = 32ticks
Difference from "a_delay" and "a_delay1" is only 1instruction[sub $C_stTOS , # 8].
Why is different?
I have no idea.

Assembler word"a_pulse" output pulse.

This is often used for bit-mask of port

```
    mov       $C_treg3 , # 1
    shl       $C_treg3 , $C_stTOS
    \ Set pins to output
    or        dira , $C_treg3
```

Word'pulse1' output Hi-pulse(10usec) on P0.
Word'pulse2' output Hi-pulse(100usec) on P0.