may-2021 (an)

RISC-V assembler notation in noForth

*For RISC-V instructions see riscv-spec.pdf on https://riscv.org
or search internet for "risc-v green card".*

1. Registers

Register names in noForth assembler

```
noForth
ZERO    X0
LINK    X1
RP       X2      return stack pointer

XX      X3      not used by noForth
YY      X4      not used by noForth
ZZ      X5      not used by noForth

RAM     X6      start of RAM section
NXT     X7      address of NEXT routine
TOS     X8      top of data stack
SP       X9      data stack pointer
IP      X10     noForth instruction pointer

W       X11     scratch reg. used by NEXT
HOP     X12     scratch reg. used by NEXT
DAY     X13     scratch reg. used by code words
SUN     X14     scratch reg. used by code words
MOON    X15     scratch reg. used by code words
```

W, HOP, DAY, SUN and MOON (X11..X15) are local scratch regeisters. NoForth uses them, but you may also use them within your code definitions. As soon as you leave the definition their value becomes uncertain.

Other registers

The registers X16..X31 are not defined and not used in noForth. It is very easy to define them if you need them:

```
8010 constant X16    (hex 8000 + register number)
801F constant X31
   etc.
```

2. Forth style assembler

The noForth RISC-V assembler is in forth style. This means:
1. First the operands, then the instruction name
2. Spaces between operands, instead of commas

```
noForth style
tos day sun ADD     \ ADD tos,day,sun
tos day 2 ADDI      \ ADDI tos,day,2
```

3. Compressed code

Drop the 'c' from compressed instruction names, the dot remains.

```
tos day .add        \ c.add tos,day
tos -1 .addi        \ c.addi tos,-1
```

4. Memory addressing

```
tos day ) .lw       \ c.lw tos,day,0
tos day ) LB        \ LB tos,day,0
day sun ) SB        \ SB day,sun,0
day 4 sun x) .sw    \ c.sw day,sun,4
tos 3 day x) LB     \ LB tos,day,3
day 1 sun x) SB     \ SB day,sun,1
```

The may-2021 version of the assembler also accepts the following shorter notation:

```
tos  day 0 .lw      \ c.lw tos,day,0
tos  day 0 LB       \ LB tos,day,0
day  sun 0 SB       \ SB day,sun,0
day  sun 4 .sw      \ c.sw day,sun,4
tos  day 3 LB       \ LB tos,day,3
day  sun 1 SB       \ SB day,sun,1
```

5. Decisions (branches)

```
.0=?  .0<>?         \ 1 operand
=?    <>?           \ 2 operands
>?    <EQ?
U>?   U<EQ?
```

Use these conditions before IF, WHILE, UNTIL,

```
tos .0=? IF, .. THEN,
sun moon <EQ? IF, .. ELSE, .. THEN,
BEGIN, .. tos .0<>? WHILE, .. REPEAT,
BEGIN, .. sun moon >? UNTIL,
AHEAD, .. THEN,
BEGIN, .. AGAIN,
```

6. .mov

The macro .mov handles .lw .sw .mw .lwsp and .swsp .
It accepts source)+ and destination -), also with RP .
For .mov the order of the operands is always 'destiny source'. This means that .mov can distinguish between load and store actions.

```
macro                  result
sun day .mov           \ sun day .mw
tos sp ) .mov          \ tos sp ) .lw
sp ) tos .mov          \ tos sp ) .sw
tos sp )+ .mov         \ tos sp ) .lw   sp 4 .addi
rp -) tos .mov         \ rp -4 .addi   tos rp ) .swsp
```

7. BMOV HMOV WMOV

The macros BMOV HMOV WMOV handle LBU SB LHU SH MW LW and SW .

```
macro                   result
tos day )+ BMOV         \ tos day ) LBU   day 1 .addi
day ) tos HMOV          \ tos day ) SH
```

8. LI

The macro LI loads any 32 bit number in a register.

```
macro                   result
tos 1234ABCD LI         \ tos 1234B000 LUI   tos tos -433 ADDI
tos 500 LI              \ tos zero 500 ADDI
tos -3 LI               \ tos -3 .li
```

9. Error messages

```
MSG from ?.REG      illegible register or register not allowed
MSG from ?REG       illegible register
MSG from ?R0        R0 not allowed
MSG from `-)`       -) not allowed
MSG from `)+`       )+ not allowed
MSG from ?RANGE.U   unsigned immediate range error
MSG from ?RANGE.S   signed immediate range error
MSG from IF,        unknown condition
MSG from THEN,      unbalanced
MSG from UNTIL,     unbalanced, or unknown condition
MSG from AGAIN,     unbalanced
```

10. noForth assembler words

```
.ADD .ADDI .AND .ANDI .JALR .JR .LI .MW .OR .SLLI .SRAI .SRLI .SUB .XOR
ADD ADDI ANDI AUIPC J LB LBU LH LHU LUI LW MRET ORI
SB SH SLL SLT SLTI SLTIU SLTU SRA SRL SUB SW WFI XORI
DIV DIVU MUL MULH MULHSU MULHU REM REMU
CSRRC CSRRCI CSRRS CSRRSI CSRRW CSRRWI
Macros: .MOV BMOV HMOV WMOV LI NEXT   Mem.addr: ) X) -) )+
Decisions: .0<>? .0=?   <>? <EQ? =? >? U<EQ? U>?
AGAIN, AHEAD, BEGIN, ELSE, IF, REPEAT, THEN, UNTIL, WHILE,
```

11. Code examples

```
noForth                     result

code DROP
    tos sp )+ .mov          tos sp ) .lw
                            sp 4 .addi
    next end-code           nxt .jr

 code 2DROP ( x y -- )
    tos 4 sp x) .mov        tos 4 sp x) .lw
    sp 8 .addi              sp 8 .addi
    next end-code           nxt .jr

code DUP ( x -- x x )
    sp -) tos .mov          sp -4 .addi
                            tos sp ) .sw
    next end-code           nxt .jr

code >R ( x -- )
    rp -) tos .mov          rp -4 .addi
                            tos rp ) .swsp
    tos sp )+ .mov          tos sp ) .lw
                            sp 4 .addi
    next end-code           nxt .jr

code >DIG ( n -- ch )
    day hx 0A li            day A .li
    day tos U<EQ?           tos day 6 BLTU
    if, tos 7 .addi         tos 7 .addi
    then,
    tos tos ch 0 ADDI       tos tos 30 ADDI
    next end-code           nxt .jr
```