Example of S2 assembly programming

```
;; ---------- while -----------------

;; while i < n
;;    i = i + 1

;; let i = r1, n = r2, t = r3

.a 0
.c
        ld r2 #10
:loop   sub r3 r1 r2      ;; i-n < 0
        jmp ge exit
        add r1 r1 #1
        jmp always loop
:exit   trap print r1
        trap stop r0
.e

;; ----------------------------------------
;;  show loop, array access

;;  global  A[10], I;

;; main
;;  I = 0;
;;  while( I < 10 )
;;    A[I] = I + 2;
;;    I = I + 1;

;; let r2 = I, r3 = temp

.s
        A  2000
.a 0
.c
        xor r2 r2 r2
:while  sub r0 r2 #10
        jmp ge exit
        add r3 r2 #2
        st @A r2 r3
        add r2 r2 #1
        jmp always while
:exit   trap stop r0
.e

;; ----------- for ---------------------

;; s = 0
;; for i=0; i<=end; i++
;;    s = s + i

;; let i=r4, end=r2, s=r3, t=r4

.a 0
.c
        ld r1 #1
        ld r2 #10
:loop   sub r4 r1 r2
        jmp gt exit
        add r3 r3 r1
        add r1 r1 #1
        jmp always loop
:exit   trap stop r0
.e

;; ---------- call -----------------------

;; sum(a,b)
;;   return a + b

;; main
;;   sum(4,5)

;; for sum
;; let a=r1, b=r2,
;; let retval=r29, link=r30, sp=r31
;; let sum_a, sum_b

.s
        sum_a 1000
        sum_b 1001
.a 0
.c
        ld r31 #2000
:main   ld r1 #4
        st sum_a r1
        ld r1 #5
        st sum_b r1
        jal r30 sum
        trap print r29
        trap stop r0

:sum    st @1 r31 r1
        st @2 r31 r2
        add r31 r31 #2  ;; push r1,r2
        ld r1 sum_a
        ld r2 sum_b     ;; pass a b
        add r29 r1 r2   ;; a + b
        ld r2 @0 r31
        ld r1 @-1 r31
        sub r31 r31 #2  ;; pop r2,r1
        jr r30
.e
```

```
;; ----------  fac -----------------

;; fac(n)
;;   if n == 0 return 1
;;   else return n * fac(n-1)

;; main
;;   fac(4)

;; let n=r1,t=r2
;; let retval=r29, link=r30, sp=r31

;; let fac_n

.s
        fac_n 1000
.a 0
.c
        ld r31 #2000
:main   ld r1 #4
        st fac_n r1
        jal r30 fac
        trap print r29
        trap stop r0

:fac    st @1 r31 r1
        st @2 r31 r2
        st @3 r31 r30
        add r31 r31 #3  ;; push r1 r2 link
        ld r1 fac_n     ;; pass n
        sub r2 r1 r0    ;; n == 0
        jmp neq else
        ld r29 #1       ;; ret 1
        jmp always ret
:else   sub r2 r1 #1
        st fac_n r2     ;; n-1
        jal r30 fac
        mul r29 r1 r29  ;; n*fac(n-1)
:ret    ld r30 @0 r31
        ld r2 @-1 r31
        ld r1 @-2 r31
        sub r31 r31 #3  ;; pop link r2 r1
        jr r30
.e

;; ----------- list ---------------------

;; linked-list
;; list
;;   data
;;   next

;; search for d in list x
;; return 1 found, 0 not found
```

```
;; search(x,d)
;;   int flag
;;   flag = 0
;;   while x != nil
;;     if x.data == d
;;       flag = 1
;;       break
;;     else
;;       x = x.next
;;   return flag

;; test list (7,8,9)
;;    1000:7, 1001:1002, 1002:8, 1003:1004,
;;    1004:9, 1005:0

;; main
;;   print search(list,8)

;; let x=r1, d=r2, flag=r3, t=r4, x.data=r5
;; let retval=r29, link=r30, sp=r31

;; let search_x, search_d, list

.s
        search_x 1010
        search_d 1011
        list 1000       ;; to 1005
.a 0
.c
        ld r31 #2000
        ld r1 #7
        st 1000 r1
        ld r1 #1002
        st 1001 r1
        ld r1 #8
        st 1002 r1
        ld r1 #1004
        st 1003 r1
        ld r1 #9
        st 1004 r1
        ld r1 #0
        st 1005 r1      ;; list (7,8,9)

:main   ld r1 #1000
        st search_x r1
        ld r1 #8
        st search_d r1
        jal r30 search
        trap print r29
        trap stop r0

:searchst @1 r31 r1
        st @2 r31 r2
        st @3 r31 r3
        st @4 r31 r4
        st @5 r31 r5
        add r31 r31 #5  ;; push r1..r5
        ld r1 search_x
        ld r2 search_d  ;; pass x d
        ld r3 #0        ;; flag=0
:loop   sub r4 r1 r0    ;; x != nil
        jmp eq ret
        ld r5 @0 r1     ;; x.data
        sub r4 r5 r2    ;; x.data == d
        jmp neq else
        ld r3 #1        ;; flag=1
        jmp always ret
:else   ld r1 @1 r1     ;; x=x.next
        jmp always loop
:ret    or r29 r3 r0    ;; return flag
        ld r5 @0 r31
        ld r4 @-1 r31
        ld r3 @-2 r31
        ld r2 @-3 r31
        ld r1 @-4 r31
        sub r31 r31 #5  ;; pop r5..r1
        jr r30
.e
```