Bilag D - Aftesting af faserne ressource til emit med eksemplerne fra imada

:

:

Start på test af test_code_sdu_AbsTest.tony

---

test_code_sdu_AbsTest.tony

```
:
001: var a : int;
002: var b : int;
003: var c : int;
004:
005: a = 0 - 7;
006: b = 6 - 8;
007: c = 6;
008:
009: write a;
010: write b;
011: write |a|;
012: write |b|;
013: write |a| + |c|;
014: write |a+c|;
```

:

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_AbsTest.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

   3 gennemloeb og reduceret med:

   5 instruktioner
```

```
   0 multiplikationer

   0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut


Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

-----------------------------------------------------
:
```

consol output fra test_code_sdu_AbsTest.exe

```
:
-7
-2
7
2
13
1
-----------------------------------------------------
:
tonyprettyTxt.txt
:
-----------------------------------------------------
:
tonyprettyXml.xml
:
-----------------------------------------------------
:
tonysymbol_xref.txt
:
-----------------------------------------------------
:
```

test_code_sdu_AbsTest.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
```

```
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope

.equ  a_1,-12  #1:int lokal adr. på ebp
.equ  b_2,-8  #2:int lokal adr. på ebp
.equ  c_3,-4  #3:int lokal adr. på ebp

_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $12,%esp  #reserver memory for lokale data
#main insructions start
#5:assign
###  movl  $0,%eax  #num
###  movl  $7,%ebx  #num
###  subl  %ebx,%eax  #-
###  subl  $7,%eax  #optimized add/sub
  movl  $-7,%eax  #optimized twize add/sub const
  movl  %eax,a_1(%ebp)  #5:Store lokal variabel

#6:assign
###  movl  $6,%eax  #num
###  movl  $8,%ebx  #num
###  subl  %ebx,%eax  #-
###  subl  $8,%eax  #optimized add/sub
  movl  $-2,%eax  #optimized twize add/sub const
  movl  %eax,b_2(%ebp)  #6:Store lokal variabel

#7:assign
  movl  $6,%eax  #7:num
  movl  %eax,c_3(%ebp)  #7:Store lokal variabel

#9:write
  movl  a_1(%ebp),%eax  #9:Load lokal variabel
  pushl  %eax
  pushl  $_int_format
  call  printf
  addl  $8,%esp  #Fjern parametre igen

#10:write
  movl  b_2(%ebp),%eax  #10:Load lokal variabel
  pushl  %eax
```

```
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#11:write
#11:Numeric
 movl a_1(%ebp),%eax #11:Load lokal variabel
 orl %eax,%eax #11:>0 ?
 jge _positiv_1
 negl %eax #11:vend fortegn
_positiv_1:
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#12:write
#12:Numeric
 movl b_2(%ebp),%eax #12:Load lokal variabel
 orl %eax,%eax #12:>0 ?
 jge _positiv_2
 negl %eax #12:vend fortegn
_positiv_2:
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#13:write
#13:Numeric
 movl a_1(%ebp),%eax #13:Load lokal variabel
 orl %eax,%eax #13:>0 ?
 jge _positiv_3
 negl %eax #13:vend fortegn
_positiv_3:
#13:Numeric
 movl c_3(%ebp),%ebx #13:Load lokal variabel
 orl %ebx,%ebx #13:>0 ?
 jge _positiv_4
 negl %ebx #13:vend fortegn
_positiv_4:
 addl %ebx,%eax #13:+
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#14:write
#14:Numeric
 movl a_1(%ebp),%eax #14:Load lokal variabel
### movl c_3(%ebp),%ebx #Load lokal variabel
### addl %ebx,%eax #+
 addl c_3(%ebp),%eax #optimized add/sub
 orl %eax,%eax #14:>0 ?
 jge _positiv_5
 negl %eax #14:vend fortegn
_positiv_5:
 pushl %eax
 pushl $_int_format
```

```
  call  printf
  addl  $8,%esp  #Fjern parametre igen

#main insructions slut
  xorl  %eax,%eax  #afslut normal med returværdi 0
main_end:
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#--------------------------
_indexunderflow:
  pushl  %eax
  pushl  $_indexunderflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_indexoverflow:
  pushl  %eax
  pushl  $_indexoverflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_zerodivide:
  pushl  %eax
  pushl  $_zerodivide_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_nonpositivelements:
  pushl  %eax
  pushl  $_nonpositivelements_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_outofmemory:
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_AbsTest.tony
```

:
::
:
:

:

Start på test af test_code_sdu_ArrayIndex.tony

---

test_code_sdu_ArrayIndex.tony

```
:
001: type intArray = array of int;
002: var a : intArray;
003:
004: new a of length 7;
005:
006: a[0] = 4;
007: a[1] = 5;
008: a[2] = 6;
009: a[3] = 7;
010: a[4] = 8;
011: a[5] = 9;
012: a[6] = 10;
013:
014: write a[0];
015: write a[6];
---
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_ArrayIndex.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

   1 gennemloeb og reduceret med:
```

```
   0 instruktioner

   0 multiplikationer

   0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut


Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

-----------------------------------------------------
:
```

consol output fra test_code_sdu_ArrayIndex.exe

```
:
4
10
-----------------------------------------------------
:
tonyprettyTxt.txt
:
-----------------------------------------------------
:
tonyprettyXml.xml
:
-----------------------------------------------------
:
tonysymbol_xref.txt
:
-----------------------------------------------------
:
```

test_code_sdu_ArrayIndex.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
```

```
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope

.equ  a_1,-4  #2:array lokal adr. på ebp

_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $4,%esp  #reserver memory for lokale data
#main insructions start
#4:new length
  movl  $7,%eax  #4:num
  orl  %eax,%eax  #test positivt antal elementer
  jg  _newLength_1_1
  movl  $4,%eax  #linienr på fejl
  jmp  _nonpositivelements
_newLength_1_1:
  pushl  %eax  #Save antal elm
  movl  $4,%ebx  #ebx = elementlaengde
  imull  %ebx  #memory_size = elementer*elementlængde
  addl  $4,%eax  #plads til antals variabel
  movl  _myheapfreeadr,%esi
  movl  %eax,0(%esi)  #gem allokeret mem size
  addl  %esi,%eax  #ny freeadr
  addl  $4,%eax  #beregnes
  movl  %eax,_myheapfreeadr  #og gemmes
  cmpl  $_aftermyheapspace,%eax  #test for out of memory
  jng  _newLength_1_2
  movl  $4,%eax  #linienr på fejl
  jmp  _outofmemory
_newLength_1_2:
  leal  8(%esi),%eax  #adresse på array : offset -8 er mem.size og -4 antal elm.
  popl  %ebx
  movl  %ebx,-4(%eax)  #Gem antal elm. i array -4
#store i variabel
  movl  %eax,a_1(%ebp)  #4:Store lokal variabel

#6:assign
  movl  $4,%eax  #6:num
#6:***** abstractAsmVariable_Indexed
#6:indexed store
  pushl  %eax  #6:save 1. op
#6:calc indexed adr.
  pushl  %ebx  #6:save
```

```
 movl a_1(%ebp),%eax #6:hent adr.
 movl $0,%ebx #6:num
 orl %ebx,%ebx #6:test for index underflow
 jge _index_2_test2
 movl $6,%eax #6:linienr på fejl
 jmp _indexunderflow
_index_2_test2:
 cmpl %ebx,-4(%eax) #6:test for index overflow = offset fejl
 jg _index_2_ok
 movl $6,%eax #6:linienr på fejl
 jmp _indexoverflow
_index_2_ok:
 pushl %eax #6:save array-adr.
 movl $4,%eax #6:hent elm. længde eax
 imull %ebx #6:eax = elm. offset
 popl %ebx #6:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #6:adr. paa array elm. adresse i eax
 popl %ebx #6:reetabler
#6:indexed load/store
 popl %edx #6:hent værdi igen
 movl %edx,0(%eax) #6:store variabel

#7:assign
 movl $5,%eax #7:num
#7:***** abstractAsmVariable_Indexed
#7:indexed store
 pushl %eax #7:save 1. op
#7:calc indexed adr.
 pushl %ebx #7:save
 movl a_1(%ebp),%eax #7:hent adr.
 movl $1,%ebx #7:num
 orl %ebx,%ebx #7:test for index underflow
 jge _index_3_test2
 movl $7,%eax #7:linienr på fejl
 jmp _indexunderflow
_index_3_test2:
 cmpl %ebx,-4(%eax) #7:test for index overflow = offset fejl
 jg _index_3_ok
 movl $7,%eax #7:linienr på fejl
 jmp _indexoverflow
_index_3_ok:
 pushl %eax #7:save array-adr.
 movl $4,%eax #7:hent elm. længde eax
 imull %ebx #7:eax = elm. offset
 popl %ebx #7:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #7:adr. paa array elm. adresse i eax
 popl %ebx #7:reetabler
#7:indexed load/store
 popl %edx #7:hent værdi igen
 movl %edx,0(%eax) #7:store variabel

#8:assign
 movl $6,%eax #8:num
#8:***** abstractAsmVariable_Indexed
#8:indexed store
 pushl %eax #8:save 1. op
#8:calc indexed adr.
 pushl %ebx #8:save
 movl a_1(%ebp),%eax #8:hent adr.
```

```
 movl $2,%ebx #8:num
 orl %ebx,%ebx #8:test for index underflow
 jge _index_4_test2
 movl $8,%eax #8:linienr på fejl
 jmp _indexunderflow
_index_4_test2:
 cmpl %ebx,-4(%eax) #8:test for index overflow = offset fejl
 jg _index_4_ok
 movl $8,%eax #8:linienr på fejl
 jmp _indexoverflow
_index_4_ok:
 pushl %eax #8:save array-adr.
 movl $4,%eax #8:hent elm. længde eax
 imull %ebx #8:eax = elm. offset
 popl %ebx #8:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #8:adr. paa array elm. adresse i eax
 popl %ebx #8:reetabler
#8:indexed load/store
 popl %edx #8:hent værdi igen
 movl %edx,0(%eax) #8:store variabel

#9:assign
 movl $7,%eax #9:num
#9:***** abstractAsmVariable_Indexed
#9:indexed store
 pushl %eax #9:save 1. op
#9:calc indexed adr.
 pushl %ebx #9:save
 movl a_1(%ebp),%eax #9:hent adr.
 movl $3,%ebx #9:num
 orl %ebx,%ebx #9:test for index underflow
 jge _index_5_test2
 movl $9,%eax #9:linienr på fejl
 jmp _indexunderflow
_index_5_test2:
 cmpl %ebx,-4(%eax) #9:test for index overflow = offset fejl
 jg _index_5_ok
 movl $9,%eax #9:linienr på fejl
 jmp _indexoverflow
_index_5_ok:
 pushl %eax #9:save array-adr.
 movl $4,%eax #9:hent elm. længde eax
 imull %ebx #9:eax = elm. offset
 popl %ebx #9:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #9:adr. paa array elm. adresse i eax
 popl %ebx #9:reetabler
#9:indexed load/store
 popl %edx #9:hent værdi igen
 movl %edx,0(%eax) #9:store variabel

#10:assign
 movl $8,%eax #10:num
#10:***** abstractAsmVariable_Indexed
#10:indexed store
 pushl %eax #10:save 1. op
#10:calc indexed adr.
 pushl %ebx #10:save
 movl a_1(%ebp),%eax #10:hent adr.
 movl $4,%ebx #10:num
```

```
 orl  %ebx,%ebx  #10:test for index underflow
 jge  _index_6_test2
 movl  $10,%eax  #10:linienr på fejl
 jmp  _indexunderflow
_index_6_test2:
 cmpl  %ebx,-4(%eax)  #10:test for index overflow = offset fejl
 jg  _index_6_ok
 movl  $10,%eax  #10:linienr på fejl
 jmp  _indexoverflow
_index_6_ok:
 pushl  %eax  #10:save array-adr.
 movl  $4,%eax  #10:hent elm. længde eax
 imull  %ebx  #10:eax = elm. offset
 popl  %ebx  #10:restore array-adr. i ebx
 leal  0(%ebx,%eax),%eax  #10:adr. paa array elm. adresse i eax
 popl  %ebx  #10:reetabler
#10:indexed load/store
 popl  %edx  #10:hent værdi igen
 movl  %edx,0(%eax)  #10:store variabel

#11:assign
 movl  $9,%eax  #11:num
#11:***** abstractAsmVariable_Indexed
#11:indexed store
 pushl  %eax  #11:save 1. op
#11:calc indexed adr.
 pushl  %ebx  #11:save
 movl  a_1(%ebp),%eax  #11:hent adr.
 movl  $5,%ebx  #11:num
 orl  %ebx,%ebx  #11:test for index underflow
 jge  _index_7_test2
 movl  $11,%eax  #11:linienr på fejl
 jmp  _indexunderflow
_index_7_test2:
 cmpl  %ebx,-4(%eax)  #11:test for index overflow = offset fejl
 jg  _index_7_ok
 movl  $11,%eax  #11:linienr på fejl
 jmp  _indexoverflow
_index_7_ok:
 pushl  %eax  #11:save array-adr.
 movl  $4,%eax  #11:hent elm. længde eax
 imull  %ebx  #11:eax = elm. offset
 popl  %ebx  #11:restore array-adr. i ebx
 leal  0(%ebx,%eax),%eax  #11:adr. paa array elm. adresse i eax
 popl  %ebx  #11:reetabler
#11:indexed load/store
 popl  %edx  #11:hent værdi igen
 movl  %edx,0(%eax)  #11:store variabel

#12:assign
 movl  $10,%eax  #12:num
#12:***** abstractAsmVariable_Indexed
#12:indexed store
 pushl  %eax  #12:save 1. op
#12:calc indexed adr.
 pushl  %ebx  #12:save
 movl  a_1(%ebp),%eax  #12:hent adr.
 movl  $6,%ebx  #12:num
 orl  %ebx,%ebx  #12:test for index underflow
```

```
 jge _index_8_test2
 movl $12,%eax #12:linienr på fejl
 jmp _indexunderflow
_index_8_test2:
 cmpl %ebx,-4(%eax) #12:test for index overflow = offset fejl
 jg _index_8_ok
 movl $12,%eax #12:linienr på fejl
 jmp _indexoverflow
_index_8_ok:
 pushl %eax #12:save array-adr.
 movl $4,%eax #12:hent elm. længde eax
 imull %ebx #12:eax = elm. offset
 popl %ebx #12:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #12:adr. paa array elm. adresse i eax
 popl %ebx #12:reetabler
#12:indexed load/store
 popl %edx #12:hent værdi igen
 movl %edx,0(%eax) #12:store variabel

#14:write
#14:***** abstractAsmVariable_Indexed
#14:indexed load
#14:calc indexed adr.
 pushl %ebx #14:save
 movl a_1(%ebp),%eax #14:hent adr.
 movl $0,%ebx #14:num
 orl %ebx,%ebx #14:test for index underflow
 jge _index_9_test2
 movl $14,%eax #14:linienr på fejl
 jmp _indexunderflow
_index_9_test2:
 cmpl %ebx,-4(%eax) #14:test for index overflow = offset fejl
 jg _index_9_ok
 movl $14,%eax #14:linienr på fejl
 jmp _indexoverflow
_index_9_ok:
 pushl %eax #14:save array-adr.
 movl $4,%eax #14:hent elm. længde eax
 imull %ebx #14:eax = elm. offset
 popl %ebx #14:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #14:adr. paa array elm. adresse i eax
 popl %ebx #14:reetabler
#14:indexed load/store
 movl 0(%eax),%eax #14:load 1. op. variabel
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#15:write
#15:***** abstractAsmVariable_Indexed
#15:indexed load
#15:calc indexed adr.
 pushl %ebx #15:save
 movl a_1(%ebp),%eax #15:hent adr.
 movl $6,%ebx #15:num
 orl %ebx,%ebx #15:test for index underflow
 jge _index_10_test2
 movl $15,%eax #15:linienr på fejl
```

```
 jmp _indexunderflow
_index_10_test2:
 cmpl %ebx,-4(%eax) #15:test for index overflow = offset fejl
 jg _index_10_ok
 movl $15,%eax #15:linienr på fejl
 jmp _indexoverflow
_index_10_ok:
 pushl %eax #15:save array-adr.
 movl $4,%eax #15:hent elm. længde eax
 imull %ebx #15:eax = elm. offset
 popl %ebx #15:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #15:adr. paa array elm. adresse i eax
 popl %ebx #15:reetabler
#15:indexed load/store
 movl 0(%eax),%eax #15:load 1. op. variabel
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#main insructions slut
 xorl %eax,%eax #afslut normal med returværdi 0
main_end:
 movl %ebp,%esp #restore stak ptr
 popl %ebp #restore caler frame-ptr
 ret
#--------------------------
_indexunderflow:
 pushl %eax
 pushl $_indexunderflow_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_indexoverflow:
 pushl %eax
 pushl $_indexoverflow_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_zerodivide:
 pushl %eax
 pushl $_zerodivide_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_nonpositivelements:
 pushl %eax
 pushl $_nonpositivelements_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit
```

```
_outofmemory:
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_ArrayIndex.tony
:
:::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-----------------------------------------------------
:
```

Start på test af test_code_sdu_Assoc.tony

---

test_code_sdu_Assoc.tony

```
:
001: write 10-2*2;
002: write 24/4*2;
003: write 100/2/5/10;
004: write 4*5-3*6;
005:
-----------------------------------------------------
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_Assoc.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start
```

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

2 gennemloeb og reduceret med:

2 instruktioner

2 multiplikationer

0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut

Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

:

consol output fra test_code_sdu_Assoc.exe

:
6
12
1
2

:
tonyprettyTxt.txt
:

:
tonyprettyXml.xml
:

:
tonysymbol_xref.txt
:

:

test_code_sdu_Assoc.s

:
Oversat fra TONY
Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:

```
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope


_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $0,%esp  #reserver memory for lokale data
#main insructions start
#1:write
  movl  $10,%eax  #1:num
  pushl  %eax  #1:gem 1. operant eax
###  movl  $2,%eax  #num
###  movl  $2,%ebx  #num
###  imull  %ebx  #*
  movl  $2,%eax  #optimized mul 1.op er 2
  addl  %eax,%eax  #optimized mul 1.op er 2
  movl  %eax,%ebx  #1:aflever i 2. operant ebx
  popl  %eax  #1:reetabeler 1. operant eax
  subl  %ebx,%eax  #1:-
  pushl  %eax
  pushl  $_int_format
  call  printf
  addl  $8,%esp  #Fjern parametre igen

#2:write
  movl  $24,%eax  #2:num
  movl  $4,%ebx  #2:num
  orl  %ebx,%ebx  #2:test for zero-divide
  jne  _divide_1
  movl  $2,%eax  #2:linienr på fejl
  jmp  _zerodivide
_divide_1:
  cltd  #Klargoer for division
```

```
 idivl %ebx #2:/
### movl $2,%ebx #num
### imull %ebx #*
 addl %eax,%eax #optimized mul 2.op er 2
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#3:write
 movl $100,%eax #3:num
 movl $2,%ebx #3:num
 orl %ebx,%ebx #3:test for zero-divide
 jne _divide_2
 movl $3,%eax #3:linienr på fejl
 jmp _zerodivide
_divide_2:
 cltd #Klargoer for division
 idivl %ebx #3:/
 movl $5,%ebx #3:num
 orl %ebx,%ebx #3:test for zero-divide
 jne _divide_3
 movl $3,%eax #3:linienr på fejl
 jmp _zerodivide
_divide_3:
 cltd #Klargoer for division
 idivl %ebx #3:/
 movl $10,%ebx #3:num
 orl %ebx,%ebx #3:test for zero-divide
 jne _divide_4
 movl $3,%eax #3:linienr på fejl
 jmp _zerodivide
_divide_4:
 cltd #Klargoer for division
 idivl %ebx #3:/
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#4:write
 movl $4,%eax #4:num
 movl $5,%ebx #4:num
 imull %ebx #4:*
 pushl %eax #4:gem 1. operant eax
 movl $3,%eax #4:num
 movl $6,%ebx #4:num
 imull %ebx #4:*
 movl %eax,%ebx #4:aflever i 2. operant ebx
 popl %eax #4:reetabeler 1. operant eax
 subl %ebx,%eax #4:-
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#main insructions slut
 xorl %eax,%eax #afslut normal med returværdi 0
main_end:
```

```
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#--------------------------
_indexunderflow:
  pushl  %eax
  pushl  $_indexunderflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_indexoverflow:
  pushl  %eax
  pushl  $_indexoverflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_zerodivide:
  pushl  %eax
  pushl  $_zerodivide_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_nonpositivelements:
  pushl  %eax
  pushl  $_nonpositivelements_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_outofmemory:
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_Assoc.tony
:
::::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-----------------------------------------------------
:
```

Start på test af test_code_sdu_ErrOutOfBounds1.tony

---

test_code_sdu_ErrOutOfBounds1.tony

```
:
001: func echo(i : int) : int
002:   return i;
003: end echo;
004:
005: type intArray = array of int;
006:
007: var myArray : intArray;
008: var dummy : int;
009:
010: new myArray of length 3;
011:
012: myArray[0] = 5;
013: myArray[1] = 6;
014: myArray[2] = 7;
015:
016: dummy = echo(0);
017: write myArray[dummy];
018: dummy = echo(1);
019: write myArray[dummy];
020: dummy = echo(2);
021: write myArray[dummy];
022: dummy = echo(3);
023: write myArray[dummy];
024:
---------------------------------------------------
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_ErrOutOfBounds1.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:
```

1 gennemloeb og reduceret med:

0 instruktioner

0 multiplikationer

0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut

Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

:

consol output fra test_code_sdu_ErrOutOfBounds1.exe

:
5
6
7
runtime fejl: index overloeb i linie 23

:
tonyprettyTxt.txt
:

:
tonyprettyXml.xml
:

:
tonysymbol_xref.txt
:

:

test_code_sdu_ErrOutOfBounds1.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
```

```
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_scopeadr_1:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 1
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope

.equ  myArray_3,-8  #7:array lokal adr. på ebp
.equ  dummy_4,-4  #8:int lokal adr. på ebp

_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $8,%esp  #reserver memory for lokale data
#main insructions start
#10:new length
  movl  $3,%eax  #10:num
  orl  %eax,%eax  #test positivt antal elementer
  jg  _newLength_1_1
  movl  $10,%eax  #linienr på fejl
  jmp  _nonpositivelements
_newLength_1_1:
  pushl  %eax  #Save antal elm
  movl  $4,%ebx  #ebx = elementlaengde
  imull  %ebx  #memory_size = elementer*elementlængde
  addl  $4,%eax  #plads til antals variabel
  movl  _myheapfreeadr,%esi
  movl  %eax,0(%esi)  #gem allokeret mem size
  addl  %esi,%eax  #ny freeadr
  addl  $4,%eax  #beregnes
  movl  %eax,_myheapfreeadr  #og gemmes
  cmpl  $_aftermyheapspace,%eax  #test for out of memory
  jng  _newLength_1_2
  movl  $10,%eax  #linienr på fejl
  jmp  _outofmemory
_newLength_1_2:
  leal  8(%esi),%eax  #adresse på array : offset -8 er mem.size og -4 antal elm.
  popl  %ebx
  movl  %ebx,-4(%eax)  #Gem antal elm. i array -4
#store i variabel
  movl  %eax,myArray_3(%ebp)  #10:Store lokal variabel
```

```
#12:assign
 movl $5,%eax #12:num
#12:***** abstractAsmVariable_Indexed
#12:indexed store
 pushl %eax #12:save 1. op
#12:calc indexed adr.
 pushl %ebx #12:save
 movl myArray_3(%ebp),%eax #12:hent adr.
 movl $0,%ebx #12:num
 orl %ebx,%ebx #12:test for index underflow
 jge _index_2_test2
 movl $12,%eax #12:linienr på fejl
 jmp _indexunderflow
_index_2_test2:
 cmpl %ebx,-4(%eax) #12:test for index overflow = offset fejl
 jg _index_2_ok
 movl $12,%eax #12:linienr på fejl
 jmp _indexoverflow
_index_2_ok:
 pushl %eax #12:save array-adr.
 movl $4,%eax #12:hent elm. længde eax
 imull %ebx #12:eax = elm. offset
 popl %ebx #12:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #12:adr. paa array elm. adresse i eax
 popl %ebx #12:reetabler
#12:indexed load/store
 popl %edx #12:hent værdi igen
 movl %edx,0(%eax) #12:store variabel

#13:assign
 movl $6,%eax #13:num
#13:***** abstractAsmVariable_Indexed
#13:indexed store
 pushl %eax #13:save 1. op
#13:calc indexed adr.
 pushl %ebx #13:save
 movl myArray_3(%ebp),%eax #13:hent adr.
 movl $1,%ebx #13:num
 orl %ebx,%ebx #13:test for index underflow
 jge _index_3_test2
 movl $13,%eax #13:linienr på fejl
 jmp _indexunderflow
_index_3_test2:
 cmpl %ebx,-4(%eax) #13:test for index overflow = offset fejl
 jg _index_3_ok
 movl $13,%eax #13:linienr på fejl
 jmp _indexoverflow
_index_3_ok:
 pushl %eax #13:save array-adr.
 movl $4,%eax #13:hent elm. længde eax
 imull %ebx #13:eax = elm. offset
 popl %ebx #13:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #13:adr. paa array elm. adresse i eax
 popl %ebx #13:reetabler
#13:indexed load/store
 popl %edx #13:hent værdi igen
 movl %edx,0(%eax) #13:store variabel

#14:assign
```

```
 movl $7,%eax #14:num
#14:***** abstractAsmVariable_Indexed
#14:indexed store
 pushl %eax #14:save 1. op
#14:calc indexed adr.
 pushl %ebx #14:save
 movl myArray_3(%ebp),%eax #14:hent adr.
 movl $2,%ebx #14:num
 orl %ebx,%ebx #14:test for index underflow
 jge _index_4_test2
 movl $14,%eax #14:linienr på fejl
 jmp _indexunderflow
_index_4_test2:
 cmpl %ebx,-4(%eax) #14:test for index overflow = offset fejl
 jg _index_4_ok
 movl $14,%eax #14:linienr på fejl
 jmp _indexoverflow
_index_4_ok:
 pushl %eax #14:save array-adr.
 movl $4,%eax #14:hent elm. længde eax
 imull %ebx #14:eax = elm. offset
 popl %ebx #14:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #14:adr. paa array elm. adresse i eax
 popl %ebx #14:reetabler
#14:indexed load/store
 popl %edx #14:hent værdi igen
 movl %edx,0(%eax) #14:store variabel

#16:assign
 pushl %edi #16:save edi
 subl $4,%esp #16:reserver memory for parameterblok
 movl %esp,%edi #16:etabler base-reference til parametre
 subl $8,%edi #16:juster som ebp efter kald
 movl $0,%eax #16:num
 movl %eax,i_2(%edi) #16:gem parameter paa stak
 call echo_1 #16:aktiver funktion
 addl $4,%esp #16:frigiv parameterblok
 popl %edi #16:restore edi
 movl %eax,dummy_4(%ebp) #16:Store lokal variabel

#17:write
#17:***** abstractAsmVariable_Indexed
#17:indexed load
#17:calc indexed adr.
 pushl %ebx #17:save
 movl myArray_3(%ebp),%eax #17:hent adr.
 movl dummy_4(%ebp),%ebx #17:Load lokal variabel
 orl %ebx,%ebx #17:test for index underflow
 jge _index_5_test2
 movl $17,%eax #17:linienr på fejl
 jmp _indexunderflow
_index_5_test2:
 cmpl %ebx,-4(%eax) #17:test for index overflow = offset fejl
 jg _index_5_ok
 movl $17,%eax #17:linienr på fejl
 jmp _indexoverflow
_index_5_ok:
 pushl %eax #17:save array-adr.
 movl $4,%eax #17:hent elm. længde eax
```

```
 imull  %ebx  #17:eax = elm. offset
 popl  %ebx  #17:restore array-adr. i ebx
 leal  0(%ebx,%eax),%eax  #17:adr. paa array elm. adresse i eax
 popl  %ebx  #17:reetabler
#17:indexed load/store
 movl  0(%eax),%eax  #17:load 1. op. variabel
 pushl  %eax
 pushl  $_int_format
 call  printf
 addl  $8,%esp  #Fjern parametre igen

#18:assign
 pushl  %edi  #18:save edi
 subl  $4,%esp  #18:reserver memory for parameterblok
 movl  %esp,%edi  #18:etabler base-reference til parametre
 subl  $8,%edi  #18:juster som ebp efter kald
 movl  $1,%eax  #18:num
 movl  %eax,i_2(%edi)  #18:gem parameter paa stak
 call  echo_1  #18:aktiver funktion
 addl  $4,%esp  #18:frigiv parameterblok
 popl  %edi  #18:restore edi
 movl  %eax,dummy_4(%ebp)  #18:Store lokal variabel

#19:write
#19:***** abstractAsmVariable_Indexed
#19:indexed load
#19:calc indexed adr.
 pushl  %ebx  #19:save
 movl  myArray_3(%ebp),%eax  #19:hent adr.
 movl  dummy_4(%ebp),%ebx  #19:Load lokal variabel
 orl  %ebx,%ebx  #19:test for index underflow
 jge  _index_6_test2
 movl  $19,%eax  #19:linienr på fejl
 jmp  _indexunderflow
_index_6_test2:
 cmpl  %ebx,-4(%eax)  #19:test for index overflow = offset fejl
 jg  _index_6_ok
 movl  $19,%eax  #19:linienr på fejl
 jmp  _indexoverflow
_index_6_ok:
 pushl  %eax  #19:save array-adr.
 movl  $4,%eax  #19:hent elm. længde eax
 imull  %ebx  #19:eax = elm. offset
 popl  %ebx  #19:restore array-adr. i ebx
 leal  0(%ebx,%eax),%eax  #19:adr. paa array elm. adresse i eax
 popl  %ebx  #19:reetabler
#19:indexed load/store
 movl  0(%eax),%eax  #19:load 1. op. variabel
 pushl  %eax
 pushl  $_int_format
 call  printf
 addl  $8,%esp  #Fjern parametre igen

#20:assign
 pushl  %edi  #20:save edi
 subl  $4,%esp  #20:reserver memory for parameterblok
 movl  %esp,%edi  #20:etabler base-reference til parametre
 subl  $8,%edi  #20:juster som ebp efter kald
 movl  $2,%eax  #20:num
```

```
  movl  %eax,i_2(%edi)  #20:gem parameter paa stak
  call  echo_1  #20:aktiver funktion
  addl  $4,%esp  #20:frigiv parameterblok
  popl  %edi  #20:restore edi
  movl  %eax,dummy_4(%ebp)  #20:Store lokal variabel

#21:write
#21:***** abstractAsmVariable_Indexed
#21:indexed load
#21:calc indexed adr.
  pushl  %ebx  #21:save
  movl  myArray_3(%ebp),%eax  #21:hent adr.
  movl  dummy_4(%ebp),%ebx  #21:Load lokal variabel
  orl  %ebx,%ebx  #21:test for index underflow
  jge  _index_7_test2
  movl  $21,%eax  #21:linienr på fejl
  jmp  _indexunderflow
_index_7_test2:
  cmpl  %ebx,-4(%eax)  #21:test for index overflow = offset fejl
  jg  _index_7_ok
  movl  $21,%eax  #21:linienr på fejl
  jmp  _indexoverflow
_index_7_ok:
  pushl  %eax  #21:save array-adr.
  movl  $4,%eax  #21:hent elm. længde eax
  imull  %ebx  #21:eax = elm. offset
  popl  %ebx  #21:restore array-adr. i ebx
  leal  0(%ebx,%eax),%eax  #21:adr. paa array elm. adresse i eax
  popl  %ebx  #21:reetabler
#21:indexed load/store
  movl  0(%eax),%eax  #21:load 1. op. variabel
  pushl  %eax
  pushl  $_int_format
  call  printf
  addl  $8,%esp  #Fjern parametre igen

#22:assign
  pushl  %edi  #22:save edi
  subl  $4,%esp  #22:reserver memory for parameterblok
  movl  %esp,%edi  #22:etabler base-reference til parametre
  subl  $8,%edi  #22:juster som ebp efter kald
  movl  $3,%eax  #22:num
  movl  %eax,i_2(%edi)  #22:gem parameter paa stak
  call  echo_1  #22:aktiver funktion
  addl  $4,%esp  #22:frigiv parameterblok
  popl  %edi  #22:restore edi
  movl  %eax,dummy_4(%ebp)  #22:Store lokal variabel

#23:write
#23:***** abstractAsmVariable_Indexed
#23:indexed load
#23:calc indexed adr.
  pushl  %ebx  #23:save
  movl  myArray_3(%ebp),%eax  #23:hent adr.
  movl  dummy_4(%ebp),%ebx  #23:Load lokal variabel
  orl  %ebx,%ebx  #23:test for index underflow
  jge  _index_8_test2
  movl  $23,%eax  #23:linienr på fejl
  jmp  _indexunderflow
```

```
_index_8_test2:
 cmpl %ebx,-4(%eax) #23:test for index overflow = offset fejl
 jg _index_8_ok
 movl $23,%eax #23:linienr på fejl
 jmp _indexoverflow
_index_8_ok:
 pushl %eax #23:save array-adr.
 movl $4,%eax #23:hent elm. længde eax
 imull %ebx #23:eax = elm. offset
 popl %ebx #23:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #23:adr. paa array elm. adresse i eax
 popl %ebx #23:reetabler
#23:indexed load/store
 movl 0(%eax),%eax #23:load 1. op. variabel
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#main insructions slut
 xorl %eax,%eax #afslut normal med returværdi 0
main_end:
 movl %ebp,%esp #restore stak ptr
 popl %ebp #restore caler frame-ptr
 ret
#--------------------------
_indexunderflow:
 pushl %eax
 pushl $_indexunderflow_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_indexoverflow:
 pushl %eax
 pushl $_indexoverflow_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_zerodivide:
 pushl %eax
 pushl $_zerodivide_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_nonpositivelements:
 pushl %eax
 pushl $_nonpositivelements_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_outofmemory:
```

```
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------

#1:#function echo_1

#Parameter adresser i dette scope
.equ  i_2,8  #1:int lokal adr. på ebp

#Data adresser defineret i dette scope

echo_1:
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_1  #gem frame ptr til andre scopes
  subl  $0,%esp  #reserver memory for lokale data
  pushl  %ebx
  pushl  %esi
  pushl  %edi
#funktion instructions start
#2:return
  movl  i_2(%ebp),%eax  #2:Load lokal variabel
  jmp  echo_1_end  #return med vaerdi i eax

#funktion instructions slut
echo_1_end:
  popl  %edi
  popl  %esi
  popl  %ebx
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#--------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_ErrOutOfBounds1.tony
:
::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-----------------------------------------------------
:
```

Start på test af test_code_sdu_ErrOutOfBounds2.tony

---

test_code_sdu_ErrOutOfBounds2.tony

:

```
001: func echo(i : int) : int
002:   return i;
003: end echo;
004:
005: type intArray = array of int;
006:
007: var myArray : intArray;
008: var dummy : int;
009:
010: new myArray of length 3;
011:
012: myArray[0] = 5;
013: myArray[1] = 6;
014: myArray[2] = 7;
015:
016: dummy = echo(0);
017: write myArray[dummy];
018: dummy = echo(1);
019: write myArray[dummy];
020: dummy = echo(2);
021: write myArray[dummy];
022: dummy = echo(-1);
023: write myArray[dummy];
024:
-----------------------------------------------------
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_ErrOutOfBounds2.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

   1 gennemloeb og reduceret med:

   0 instruktioner

   0 multiplikationer
```

```
   0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut


Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

-----------------------------------------------------
:
```

consol output fra test_code_sdu_ErrOutOfBounds2.exe

```
:
5
6
7
runtime fejl: index underloeb i linie 23
-----------------------------------------------------
:
tonyprettyTxt.txt
:
-----------------------------------------------------
:
tonyprettyXml.xml
:
-----------------------------------------------------
:
tonysymbol_xref.txt
:
-----------------------------------------------------
:
```

test_code_sdu_ErrOutOfBounds2.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
```

```
_scopeadr_1:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 1
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope

.equ  myArray_3,-8  #7:array lokal adr. på ebp
.equ  dummy_4,-4  #8:int lokal adr. på ebp

_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $8,%esp  #reserver memory for lokale data
#main insructions start
#10:new length
  movl  $3,%eax  #10:num
  orl  %eax,%eax  #test positivt antal elementer
  jg  _newLength_1_1
  movl  $10,%eax  #linienr på fejl
  jmp  _nonpositivelements
_newLength_1_1:
  pushl  %eax  #Save antal elm
  movl  $4,%ebx  #ebx = elementlaengde
  imull  %ebx  #memory_size = elementer*elementlængde
  addl  $4,%eax  #plads til antals variabel
  movl  _myheapfreeadr,%esi
  movl  %eax,0(%esi)  #gem allokeret mem size
  addl  %esi,%eax  #ny freeadr
  addl  $4,%eax  #beregnes
  movl  %eax,_myheapfreeadr  #og gemmes
  cmpl  $_aftermyheapspace,%eax  #test for out of memory
  jng  _newLength_1_2
  movl  $10,%eax  #linienr på fejl
  jmp  _outofmemory
_newLength_1_2:
  leal  8(%esi),%eax  #adresse på array : offset -8 er mem.size og -4 antal elm.
  popl  %ebx
  movl  %ebx,-4(%eax)  #Gem antal elm. i array -4
#store i variabel
  movl  %eax,myArray_3(%ebp)  #10:Store lokal variabel

#12:assign
  movl  $5,%eax  #12:num
#12:***** abstractAsmVariable_Indexed
#12:indexed store
  pushl  %eax  #12:save 1. op
#12:calc indexed adr.
```

```
 pushl %ebx #12:save
 movl myArray_3(%ebp),%eax #12:hent adr.
 movl $0,%ebx #12:num
 orl %ebx,%ebx #12:test for index underflow
 jge _index_2_test2
 movl $12,%eax #12:linienr på fejl
 jmp _indexunderflow
_index_2_test2:
 cmpl %ebx,-4(%eax) #12:test for index overflow = offset fejl
 jg _index_2_ok
 movl $12,%eax #12:linienr på fejl
 jmp _indexoverflow
_index_2_ok:
 pushl %eax #12:save array-adr.
 movl $4,%eax #12:hent elm. længde eax
 imull %ebx #12:eax = elm. offset
 popl %ebx #12:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #12:adr. paa array elm. adresse i eax
 popl %ebx #12:reetabler
#12:indexed load/store
 popl %edx #12:hent værdi igen
 movl %edx,0(%eax) #12:store variabel

#13:assign
 movl $6,%eax #13:num
#13:***** abstractAsmVariable_Indexed
#13:indexed store
 pushl %eax #13:save 1. op
#13:calc indexed adr.
 pushl %ebx #13:save
 movl myArray_3(%ebp),%eax #13:hent adr.
 movl $1,%ebx #13:num
 orl %ebx,%ebx #13:test for index underflow
 jge _index_3_test2
 movl $13,%eax #13:linienr på fejl
 jmp _indexunderflow
_index_3_test2:
 cmpl %ebx,-4(%eax) #13:test for index overflow = offset fejl
 jg _index_3_ok
 movl $13,%eax #13:linienr på fejl
 jmp _indexoverflow
_index_3_ok:
 pushl %eax #13:save array-adr.
 movl $4,%eax #13:hent elm. længde eax
 imull %ebx #13:eax = elm. offset
 popl %ebx #13:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #13:adr. paa array elm. adresse i eax
 popl %ebx #13:reetabler
#13:indexed load/store
 popl %edx #13:hent værdi igen
 movl %edx,0(%eax) #13:store variabel

#14:assign
 movl $7,%eax #14:num
#14:***** abstractAsmVariable_Indexed
#14:indexed store
 pushl %eax #14:save 1. op
#14:calc indexed adr.
 pushl %ebx #14:save
```

```
 movl  myArray_3(%ebp),%eax  #14:hent adr.
 movl  $2,%ebx  #14:num
 orl  %ebx,%ebx  #14:test for index underflow
 jge  _index_4_test2
 movl  $14,%eax  #14:linienr på fejl
 jmp  _indexunderflow
_index_4_test2:
 cmpl  %ebx,-4(%eax)  #14:test for index overflow = offset fejl
 jg  _index_4_ok
 movl  $14,%eax  #14:linienr på fejl
 jmp  _indexoverflow
_index_4_ok:
 pushl  %eax  #14:save array-adr.
 movl  $4,%eax  #14:hent elm. længde eax
 imull  %ebx  #14:eax = elm. offset
 popl  %ebx  #14:restore array-adr. i ebx
 leal  0(%ebx,%eax),%eax  #14:adr. paa array elm. adresse i eax
 popl  %ebx  #14:reetabler
#14:indexed load/store
 popl  %edx  #14:hent værdi igen
 movl  %edx,0(%eax)  #14:store variabel

#16:assign
 pushl  %edi  #16:save edi
 subl  $4,%esp  #16:reserver memory for parameterblok
 movl  %esp,%edi  #16:etabler base-reference til parametre
 subl  $8,%edi  #16:juster som ebp efter kald
 movl  $0,%eax  #16:num
 movl  %eax,i_2(%edi)  #16:gem parameter paa stak
 call  echo_1  #16:aktiver funktion
 addl  $4,%esp  #16:frigiv parameterblok
 popl  %edi  #16:restore edi
 movl  %eax,dummy_4(%ebp)  #16:Store lokal variabel

#17:write
#17:***** abstractAsmVariable_Indexed
#17:indexed load
#17:calc indexed adr.
 pushl  %ebx  #17:save
 movl  myArray_3(%ebp),%eax  #17:hent adr.
 movl  dummy_4(%ebp),%ebx  #17:Load lokal variabel
 orl  %ebx,%ebx  #17:test for index underflow
 jge  _index_5_test2
 movl  $17,%eax  #17:linienr på fejl
 jmp  _indexunderflow
_index_5_test2:
 cmpl  %ebx,-4(%eax)  #17:test for index overflow = offset fejl
 jg  _index_5_ok
 movl  $17,%eax  #17:linienr på fejl
 jmp  _indexoverflow
_index_5_ok:
 pushl  %eax  #17:save array-adr.
 movl  $4,%eax  #17:hent elm. længde eax
 imull  %ebx  #17:eax = elm. offset
 popl  %ebx  #17:restore array-adr. i ebx
 leal  0(%ebx,%eax),%eax  #17:adr. paa array elm. adresse i eax
 popl  %ebx  #17:reetabler
#17:indexed load/store
 movl  0(%eax),%eax  #17:load 1. op. variabel
```

```
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#18:assign
 pushl %edi #18:save edi
 subl $4,%esp #18:reserver memory for parameterblok
 movl %esp,%edi #18:etabler base-reference til parametre
 subl $8,%edi #18:juster som ebp efter kald
 movl $1,%eax #18:num
 movl %eax,i_2(%edi) #18:gem parameter paa stak
 call echo_1 #18:aktiver funktion
 addl $4,%esp #18:frigiv parameterblok
 popl %edi #18:restore edi
 movl %eax,dummy_4(%ebp) #18:Store lokal variabel

#19:write
#19:***** abstractAsmVariable_Indexed
#19:indexed load
#19:calc indexed adr.
 pushl %ebx #19:save
 movl myArray_3(%ebp),%eax #19:hent adr.
 movl dummy_4(%ebp),%ebx #19:Load lokal variabel
 orl %ebx,%ebx #19:test for index underflow
 jge _index_6_test2
 movl $19,%eax #19:linienr på fejl
 jmp _indexunderflow
_index_6_test2:
 cmpl %ebx,-4(%eax) #19:test for index overflow = offset fejl
 jg _index_6_ok
 movl $19,%eax #19:linienr på fejl
 jmp _indexoverflow
_index_6_ok:
 pushl %eax #19:save array-adr.
 movl $4,%eax #19:hent elm. længde eax
 imull %ebx #19:eax = elm. offset
 popl %ebx #19:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #19:adr. paa array elm. adresse i eax
 popl %ebx #19:reetabler
#19:indexed load/store
 movl 0(%eax),%eax #19:load 1. op. variabel
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#20:assign
 pushl %edi #20:save edi
 subl $4,%esp #20:reserver memory for parameterblok
 movl %esp,%edi #20:etabler base-reference til parametre
 subl $8,%edi #20:juster som ebp efter kald
 movl $2,%eax #20:num
 movl %eax,i_2(%edi) #20:gem parameter paa stak
 call echo_1 #20:aktiver funktion
 addl $4,%esp #20:frigiv parameterblok
 popl %edi #20:restore edi
 movl %eax,dummy_4(%ebp) #20:Store lokal variabel
```

```
#21:write
#21:***** abstractAsmVariable_Indexed
#21:indexed load
#21:calc indexed adr.
 pushl %ebx #21:save
 movl myArray_3(%ebp),%eax #21:hent adr.
 movl dummy_4(%ebp),%ebx #21:Load lokal variabel
 orl %ebx,%ebx #21:test for index underflow
 jge _index_7_test2
 movl $21,%eax #21:linienr på fejl
 jmp _indexunderflow
_index_7_test2:
 cmpl %ebx,-4(%eax) #21:test for index overflow = offset fejl
 jg _index_7_ok
 movl $21,%eax #21:linienr på fejl
 jmp _indexoverflow
_index_7_ok:
 pushl %eax #21:save array-adr.
 movl $4,%eax #21:hent elm. længde eax
 imull %ebx #21:eax = elm. offset
 popl %ebx #21:restore array-adr. i ebx
 leal 0(%ebx,%eax),%eax #21:adr. paa array elm. adresse i eax
 popl %ebx #21:reetabler
#21:indexed load/store
 movl 0(%eax),%eax #21:load 1. op. variabel
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#22:assign
 pushl %edi #22:save edi
 subl $4,%esp #22:reserver memory for parameterblok
 movl %esp,%edi #22:etabler base-reference til parametre
 subl $8,%edi #22:juster som ebp efter kald
 movl $-1,%eax #22:num
 movl %eax,i_2(%edi) #22:gem parameter paa stak
 call echo_1 #22:aktiver funktion
 addl $4,%esp #22:frigiv parameterblok
 popl %edi #22:restore edi
 movl %eax,dummy_4(%ebp) #22:Store lokal variabel

#23:write
#23:***** abstractAsmVariable_Indexed
#23:indexed load
#23:calc indexed adr.
 pushl %ebx #23:save
 movl myArray_3(%ebp),%eax #23:hent adr.
 movl dummy_4(%ebp),%ebx #23:Load lokal variabel
 orl %ebx,%ebx #23:test for index underflow
 jge _index_8_test2
 movl $23,%eax #23:linienr på fejl
 jmp _indexunderflow
_index_8_test2:
 cmpl %ebx,-4(%eax) #23:test for index overflow = offset fejl
 jg _index_8_ok
 movl $23,%eax #23:linienr på fejl
 jmp _indexoverflow
_index_8_ok:
```

```
  pushl  %eax  #23:save array-adr.
  movl  $4,%eax  #23:hent elm. længde eax
  imull  %ebx  #23:eax = elm. offset
  popl  %ebx  #23:restore array-adr. i ebx
  leal  0(%ebx,%eax),%eax  #23:adr. paa array elm. adresse i eax
  popl  %ebx  #23:reetabler
#23:indexed load/store
  movl  0(%eax),%eax  #23:load 1. op. variabel
  pushl  %eax
  pushl  $_int_format
  call  printf
  addl  $8,%esp  #Fjern parametre igen

#main insructions slut
  xorl  %eax,%eax  #afslut normal med returværdi 0
main_end:
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#---------------------------
_indexunderflow:
  pushl  %eax
  pushl  $_indexunderflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_indexoverflow:
  pushl  %eax
  pushl  $_indexoverflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_zerodivide:
  pushl  %eax
  pushl  $_zerodivide_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_nonpositivelements:
  pushl  %eax
  pushl  $_nonpositivelements_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_outofmemory:
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit
```

```
_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------

#1:#function echo_1

#Parameter adresser i dette scope
.equ  i_2,8  #1:int lokal adr. på ebp

#Data adresser defineret i dette scope

echo_1:
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_1  #gem frame ptr til andre scopes
  subl  $0,%esp  #reserver memory for lokale data
  pushl  %ebx
  pushl  %esi
  pushl  %edi
#funktion instructions start
#2:return
  movl  i_2(%ebp),%eax  #2:Load lokal variabel
  jmp  echo_1_end  #return med vaerdi i eax

#funktion instructions slut
echo_1_end:
  popl  %edi
  popl  %esi
  popl  %ebx
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#--------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_ErrOutOfBounds2.tony
:
:::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-----------------------------------------------------
:
```

Start på test af test_code_sdu_Factorial.tony

---

test_code_sdu_Factorial.tony

```
:
001: func factorial(n: int): int
002:    if (n == 0) || (n == 1) then
003:       return 1;
004:    else
005:       return n * factorial(n-1);
006: end factorial;
```

```
007:
008: write factorial(5);
009:
-----------------------------------------------------
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_Factorial.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

   2 gennemloeb og reduceret med:

   3 instruktioner

   0 multiplikationer

   0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut


Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

-----------------------------------------------------
:
```

consol output fra test_code_sdu_Factorial.exe

```
:
120
-----------------------------------------------------
:
```

```
tonyprettyTxt.txt
:
-------------------------------------------------
:
tonyprettyXml.xml
:
-------------------------------------------------
:
tonysymbol_xref.txt
:
-------------------------------------------------
:
```

test_code_sdu_Factorial.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_scopeadr_1:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 1
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope


_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $0,%esp  #reserver memory for lokale data
#main insructions start
```

```
#8:write
 pushl %edi #8:save edi
 subl $4,%esp #8:reserver memory for parameterblok
 movl %esp,%edi #8:etabler base-reference til parametre
 subl $8,%edi #8:juster som ebp efter kald
 movl $5,%eax #8:num
 movl %eax,n_2(%edi) #8:gem parameter paa stak
 call factorial_1 #8:aktiver funktion
 addl $4,%esp #8:frigiv parameterblok
 popl %edi #8:restore edi
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#main insructions slut
 xorl %eax,%eax #afslut normal med returværdi 0
main_end:
 movl %ebp,%esp #restore stak ptr
 popl %ebp #restore caler frame-ptr
 ret
#--------------------------
_indexunderflow:
 pushl %eax
 pushl $_indexunderflow_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_indexoverflow:
 pushl %eax
 pushl $_indexoverflow_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_zerodivide:
 pushl %eax
 pushl $_zerodivide_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_nonpositivelements:
 pushl %eax
 pushl $_nonpositivelements_format
 call printf
 addl $8,%esp #Ryd op paa stak
 movl $2,%eax #Returvaerdi 2
 jmp _exit

_outofmemory:
 pushl %eax
 pushl $_outofmemory_format
 call printf
 addl $8,%esp #Ryd op paa stak
```

```
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------

#1:#function factorial_1

#Parameter adresser i dette scope
.equ  n_2,8  #1:int lokal adr. på ebp

#Data adresser defineret i dette scope

factorial_1:
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_1  #gem frame ptr til andre scopes
  subl  $0,%esp  #reserver memory for lokale data
  pushl  %ebx
  pushl  %esi
  pushl  %edi
#funktion instructions start
#5:if-then-else
  movl  n_2(%ebp),%eax  #2:Load lokal variabel
###  movl  $0,%ebx  #num
###  cmpl  %ebx,%eax  #sammenlignings exp
  cmpl  $0,%eax  #optimized cmp
  je  _equation_2_true
  xorl  %eax,%eax  #2:exp false
  jmp  _equation_2_end
_equation_2_true:
  movl  $1,%eax  #2:exp true
_equation_2_end:
  pushl  %eax  #2:gem 1. operant eax
  movl  n_2(%ebp),%eax  #2:Load lokal variabel
###  movl  $1,%ebx  #num
###  cmpl  %ebx,%eax  #sammenlignings exp
  cmpl  $1,%eax  #optimized cmp
  je  _equation_3_true
  xorl  %eax,%eax  #2:exp false
  jmp  _equation_3_end
_equation_3_true:
  movl  $1,%eax  #2:exp true
_equation_3_end:
  movl  %eax,%ebx  #2:aflever i 2. operant ebx
  popl  %eax  #2:reetabeler 1. operant eax
  orl  %ebx,%eax  #2:||
  orl  %eax,%eax  #test for false
  je  _if_1_else  #false -> hop

#3:return
  movl  $1,%eax  #3:num
  jmp  factorial_1_end  #return med vaerdi i eax

  jmp  _if_1_end  #slut if
```

```
_if_1_else:

#5:return
  movl  n_2(%ebp),%eax  #5:Load lokal variabel
  pushl  %eax  #5:gem 1. operant eax
  pushl  %edi  #5:save edi
  subl  $4,%esp  #5:reserver memory for parameterblok
  movl  %esp,%edi  #5:etabler base-reference til parametre
  subl  $8,%edi  #5:juster som ebp efter kald
  movl  n_2(%ebp),%eax  #5:Load lokal variabel
###  movl  $1,%ebx  #num
###  subl  %ebx,%eax  #-
  decl  %eax  #optimized add/sub
  movl  %eax,n_2(%edi)  #5:gem parameter paa stak
  call  factorial_1  #5:aktiver funktion
  addl  $4,%esp  #5:frigiv parameterblok
  popl  %edi  #5:restore edi
  movl  %eax,%ebx  #5:aflever i 2. operant ebx
  popl  %eax  #5:reetabeler 1. operant eax
  imull  %ebx  #5:*
  jmp  factorial_1_end  #return med vaerdi i eax

_if_1_end:

#funktion instructions slut
factorial_1_end:
  popl  %edi
  popl  %esi
  popl  %ebx
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#---------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_Factorial.tony
:
:::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-----------------------------------------------------
:
```

Start på test af test_code_sdu_IfThen.tony

---

test_code_sdu_IfThen.tony

```
:
001: var b: bool;
002:
003: b = true;
004: if b then
005:    write 0;
006: else
007:    write 1;
008:
009: b = !b;
010: if b then
```

```
011:    write 1;
012: else
013:    write 0;
014:
-----------------------------------------------------
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_IfThen.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

   2 gennemloeb og reduceret med:

   2 instruktioner

   0 multiplikationer

   0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut


Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

-----------------------------------------------------
:
```

consol output fra test_code_sdu_IfThen.exe

```
:
0
0
```

```
-------------------------------------------------
:
tonyprettyTxt.txt
:
-------------------------------------------------
:
tonyprettyXml.xml
:
-------------------------------------------------
:
tonysymbol_xref.txt
:
-------------------------------------------------
:
```

test_code_sdu_IfThen.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope

.equ  b_1,-4  #1:bool lokal adr. på ebp

_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $4,%esp  #reserver memory for lokale data
```

```
#main insructions start
#3:assign
 movl $1,%eax #3:true
 movl %eax,b_1(%ebp) #3:Store lokal variabel

#7:if-then-else
### movl b_1(%ebp),%eax #Load lokal variabel
 orl %eax,%eax #test for false
 je _if_1_else #false -> hop

#5:write
 movl $0,%eax #5:num
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

 jmp _if_1_end #slut if
_if_1_else:

#7:write
 movl $1,%eax #7:num
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

_if_1_end:

#9:assign
 movl b_1(%ebp),%eax #9:Load lokal variabel
 xorl $1,%eax #9:>0 ?
 movl %eax,b_1(%ebp) #9:Store lokal variabel

#13:if-then-else
### movl b_1(%ebp),%eax #Load lokal variabel
 orl %eax,%eax #test for false
 je _if_2_else #false -> hop

#11:write
 movl $1,%eax #11:num
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

 jmp _if_2_end #slut if
_if_2_else:

#13:write
 movl $0,%eax #13:num
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

_if_2_end:

#main insructions slut
```

```
  xorl  %eax,%eax  #afslut normal med returværdi 0
main_end:
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#--------------------------
_indexunderflow:
  pushl  %eax
  pushl  $_indexunderflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_indexoverflow:
  pushl  %eax
  pushl  $_indexoverflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_zerodivide:
  pushl  %eax
  pushl  $_zerodivide_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_nonpositivelements:
  pushl  %eax
  pushl  $_nonpositivelements_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_outofmemory:
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_IfThen.tony
:
::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
```

:

Start på test af test_code_sdu_Knapsack.tony

---
test_code_sdu_Knapsack.tony
Udeladt her p.g.a omfanget

:

tony.exe consol output

:
Options:

-msg sat til test_code_sdu_Knapsack.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

2 gennemloeb og reduceret med:

36 instruktioner

0 multiplikationer

0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut

Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

:

consol output fra test_code_sdu_Knapsack.exe

```
:
50
369
600
-------------------------------------------------
:
tonyprettyTxt.txt
:
-------------------------------------------------
:
tonyprettyXml.xml
:
-------------------------------------------------
:
tonysymbol_xref.txt
:
-------------------------------------------------
:
test_code_sdu_Knapsack.s
:
Udeladt p.g.a. omfanget
-------------------------------------------------
Slut på test af test_code_sdu_Knapsack.tony
:
:::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-------------------------------------------------
:
```

Start på test af test_code_sdu_SimpleRecord.tony

```
-------------------------------------------------
```

test_code_sdu_SimpleRecord.tony

```
:
001:
002: type recordType = record of {
003:                     x: int,
004:                     y: int
005:                   };
006: var p : recordType;
007:
008: new p;
009: p.x = 1;
010: p.y = 2;
011: write p.x;
012: write p.y;
013: write p.x + p.y;
-------------------------------------------------
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_SimpleRecord.consol.txt
```

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

1 gennemloeb og reduceret med:

0 instruktioner

0 multiplikationer

0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut

Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

:

consol output fra test_code_sdu_SimpleRecord.exe

:
1
2
3

:
tonyprettyTxt.txt
:

:
tonyprettyXml.xml
:

:
tonysymbol_xref.txt
:

:

test_code_sdu_SimpleRecord.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope

#5:type recordType
#4:record of
.equ  x_1,0  #3:int indirekte adresse
.equ  y_2,4  #4:int indirekte adresse
#4:end of record
#5:end type recordType
.equ  p_3,-4  #6:record lokal adr. på ebp

_main:  #entry for windows
main:  #entry for linux
  pushl  %ebp  #save ebp
  movl  %esp,%ebp  #set frame ptr
  movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
  subl  $4,%esp  #reserver memory for lokale data
#main insructions start
#8:new
  movl  $8,%ebx  #edx = recordlaengde
  movl  _myheapfreeadr,%eax
  movl  %ebx,0(%eax)  #gem allokeret mem size
  addl  %ebx,%eax  #ny freeadr
  addl  $4,%eax  #+ 4 byte til size
```

```
 movl %eax,_myheapfreeadr #gemmes
 cmpl $_aftermyheapspace,%eax #test for out of memory
 jng _new_1
 movl $8,%eax #linienr på fejl
 jmp _outofmemory
_new_1:
 subl $8,%eax #adresse på record = freeadr - size
#store i variabel
 movl %eax,p_3(%ebp) #8:Store lokal variabel

#9:assign
 movl $1,%eax #9:num
#9:struct store
 pushl %eax #9:save 1. op
 movl p_3(%ebp),%eax #9:hent adr.
#struct load/store
 popl %edx #9:hent værdi igen
 movl %edx,x_1(%eax) #9:base adr. på struktur-variabel i eax

#10:assign
 movl $2,%eax #10:num
#10:struct store
 pushl %eax #10:save 1. op
 movl p_3(%ebp),%eax #10:hent adr.
#struct load/store
 popl %edx #10:hent værdi igen
 movl %edx,y_2(%eax) #10:base adr. på struktur-variabel i eax

#11:write
#11:struct load
 movl p_3(%ebp),%eax #11:hent adr.
#struct load/store
 movl x_1(%eax),%eax #11:load 1. op. fra base adr. på struktur-variabel i eax
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#12:write
#12:struct load
 movl p_3(%ebp),%eax #12:hent adr.
#struct load/store
 movl y_2(%eax),%eax #12:load 1. op. fra base adr. på struktur-variabel i eax
 pushl %eax
 pushl $_int_format
 call printf
 addl $8,%esp #Fjern parametre igen

#13:write
#13:struct load
 movl p_3(%ebp),%eax #13:hent adr.
#struct load/store
 movl x_1(%eax),%eax #13:load 1. op. fra base adr. på struktur-variabel i eax
#13:struct load
 pushl %eax #13:save 1. op
 movl p_3(%ebp),%eax #13:hent adr.
#struct load/store
 movl y_2(%eax),%ebx #13:load 2. op. fra base adr. på struktur-variabel i eax
 popl %eax #13:hent 1. op værdi igen
```

```
  addl  %ebx,%eax  #13:+
  pushl  %eax
  pushl  $_int_format
  call  printf
  addl  $8,%esp  #Fjern parametre igen

#main insructions slut
  xorl  %eax,%eax  #afslut normal med returværdi 0
main_end:
  movl  %ebp,%esp  #restore stak ptr
  popl  %ebp  #restore caler frame-ptr
  ret
#--------------------------
_indexunderflow:
  pushl  %eax
  pushl  $_indexunderflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_indexoverflow:
  pushl  %eax
  pushl  $_indexoverflow_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_zerodivide:
  pushl  %eax
  pushl  $_zerodivide_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_nonpositivelements:
  pushl  %eax
  pushl  $_nonpositivelements_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_outofmemory:
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#--------------------------
```

```
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_SimpleRecord.tony
:
:::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-----------------------------------------------------
:
Start på test af ttest_code_sdu_StaticLink.tony
-----------------------------------------------------
ttest_code_sdu_StaticLink.tony
:
-----------------------------------------------------
:
tony.exe consol output
:
-----------------------------------------------------
:
consol output fra ttest_code_sdu_StaticLink.exe
:
-----------------------------------------------------
:
tonyprettyTxt.txt
:
-----------------------------------------------------
:
tonyprettyXml.xml
:
-----------------------------------------------------
:
tonysymbol_xref.txt
:
-----------------------------------------------------
:
ttest_code_sdu_StaticLink.s
:
-----------------------------------------------------
Slut på test af ttest_code_sdu_StaticLink.tony
:
:::::::::::::::::::::::::::::::::::::::::::::::::::::
:
:
-----------------------------------------------------
:
```

Start på test af test_code_sdu_WhileDo.tony

```
-----------------------------------------------------
```

test_code_sdu_WhileDo.tony

```
:
001: var i: int;
002:
003: i = 3;
004: while i < 3 do {
005:    write 9;
006:    i = i + 1;
007: }
```

```
008: write 2;
009:
010: while i <= 4 do {
011:    write i;
012:    i = i + 1;
013: }
014: write 2;
015:
-----------------------------------------------------
:
```

tony.exe consol output

```
:
Options:

-msg sat til test_code_sdu_WhileDo.consol.txt

TONY Parser afsluttet normal

TONY weeding fase start
TONY weeding fase slut
TONY opbygning af symboltabel start
TONY opbygning af symboltabel slut
TONY type-check start

TONY type-check slut

TONY offset beregninger start
TONY offset beregninger slut
TONY kodegennerering af abstract assembler start

TONY kodegennerering af abstract assembler slut

TONY peep-hole optimimering start

TONY peep-hole optimimering slut:

   2 gennemloeb og reduceret med:

   4 instruktioner

   0 multiplikationer

   0 jumps

TONY assembler udskrives til <stdout> start
TONY assembler udskrives til <stdout> slut


Antal alvorligFejl: 0

Antal fejl: 0

Antal advarsler: 0

-----------------------------------------------------
:
```

consol output fra test_code_sdu_WhileDo.exe

```
:
2
3
4
2
-------------------------------------------------
:
tonyprettyTxt.txt
:
-------------------------------------------------
:
tonyprettyXml.xml
:
-------------------------------------------------
:
tonysymbol_xref.txt
:
-------------------------------------------------
:
```

test_code_sdu_WhileDo.s

```
:
# Oversat fra TONY
# Compiler skrevet af Bjørk Boye Busch - bjbu@tietgen.dk

.data

_int_format:
.ascii "%d\n\0"
_zerodivide_format:
.ascii  "runtime fejl: nul-division i linie %d\n\0"
_outofmemory_format:
.ascii  "runtime fejl: out of memory i linie %d - heapsize(10000000)\n\0"
_nonpositivelements_format:
.ascii  "runtime fejl: array <= 0 elementer i linie %d\n\0"
_indexunderflow_format:
.ascii  "runtime fejl: index underloeb i linie %d\n\0"
_indexoverflow_format:
.ascii  "runtime fejl: index overloeb i linie %d\n\0"
.align  4
_scopeadr_0:  #frame-adr. til scopelevel
.long -1  #adr. til lokale data paa scope-level 0
_myheapfreeadr:
.long  _myheapspace  #adresse på ledig heap-hukommelse
_myheapspace:
  .space  10000000  #heap-space
_aftermyheapspace:
.long  0  #4 byte ekstra da længde gemmes før test

.text

.globl _main  ##entry for windows
.globl main  ##entry for linux

#Data adresser defineret i dette scope

.equ  i_1,-4  #1:int lokal adr. på ebp

_main:  #entry for windows
```

```
main:  #entry for linux
 pushl  %ebp  #save ebp
 movl  %esp,%ebp  #set frame ptr
 movl  %ebp,_scopeadr_0  #gem frame ptr til andre scopes
 subl  $4,%esp  #reserver memory for lokale data
#main insructions start
#3:assign
 movl  $3,%eax  #3:num
 movl  %eax,i_1(%ebp)  #3:Store lokal variabel

#7:while
_while_1:
 movl  i_1(%ebp),%eax  #4:Load lokal variabel
###  movl  $3,%ebx  #num
###  cmpl  %ebx,%eax  #sammenlignings exp
 cmpl  $3,%eax  #optimized cmp
 jl  _equation_2_true
 xorl  %eax,%eax  #4:exp false
 jmp  _equation_2_end
_equation_2_true:
 movl  $1,%eax  #4:exp true
_equation_2_end:
 orl  %eax,%eax  #test for false
 je  _while_1_end  #false -> hop
#5:write
 movl  $9,%eax  #5:num
 pushl  %eax
 pushl  $_int_format
 call  printf
 addl  $8,%esp  #Fjern parametre igen

#6:assign
 movl  i_1(%ebp),%eax  #6:Load lokal variabel
###  movl  $1,%ebx  #num
###  addl  %ebx,%eax  #+
 incl  %eax  #optimized add/sub
 movl  %eax,i_1(%ebp)  #6:Store lokal variabel

 jmp  _while_1
_while_1_end:

#8:write
 movl  $2,%eax  #8:num
 pushl  %eax
 pushl  $_int_format
 call  printf
 addl  $8,%esp  #Fjern parametre igen

#13:while
_while_3:
 movl  i_1(%ebp),%eax  #10:Load lokal variabel
###  movl  $4,%ebx  #num
###  cmpl  %ebx,%eax  #sammenlignings exp
 cmpl  $4,%eax  #optimized cmp
 jle  _equation_4_true
 xorl  %eax,%eax  #10:exp false
 jmp  _equation_4_end
_equation_4_true:
 movl  $1,%eax  #10:exp true
```

```
_equation_4_end:
 orl  %eax,%eax  #test for false
 je  _while_3_end  #false -> hop
#11:write
 movl  i_1(%ebp),%eax  #11:Load lokal variabel
 pushl  %eax
 pushl  $_int_format
 call  printf
 addl  $8,%esp  #Fjern parametre igen

#12:assign
 movl  i_1(%ebp),%eax  #12:Load lokal variabel
###  movl  $1,%ebx  #num
###  addl  %ebx,%eax  #+
 incl  %eax  #optimized add/sub
 movl  %eax,i_1(%ebp)  #12:Store lokal variabel

 jmp  _while_3
_while_3_end:

#14:write
 movl  $2,%eax  #14:num
 pushl  %eax
 pushl  $_int_format
 call  printf
 addl  $8,%esp  #Fjern parametre igen

#main insructions slut
 xorl  %eax,%eax  #afslut normal med returværdi 0
main_end:
 movl  %ebp,%esp  #restore stak ptr
 popl  %ebp  #restore caler frame-ptr
 ret
#--------------------------
_indexunderflow:
 pushl  %eax
 pushl  $_indexunderflow_format
 call  printf
 addl  $8,%esp  #Ryd op paa stak
 movl  $2,%eax  #Returvaerdi 2
 jmp  _exit

_indexoverflow:
 pushl  %eax
 pushl  $_indexoverflow_format
 call  printf
 addl  $8,%esp  #Ryd op paa stak
 movl  $2,%eax  #Returvaerdi 2
 jmp  _exit

_zerodivide:
 pushl  %eax
 pushl  $_zerodivide_format
 call  printf
 addl  $8,%esp  #Ryd op paa stak
 movl  $2,%eax  #Returvaerdi 2
 jmp  _exit

_nonpositivelements:
```

```
  pushl  %eax
  pushl  $_nonpositivelements_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_outofmemory:
  pushl  %eax
  pushl  $_outofmemory_format
  call  printf
  addl  $8,%esp  #Ryd op paa stak
  movl  $2,%eax  #Returvaerdi 2
  jmp  _exit

_exit:
  movl  _scopeadr_0,%ebp  #restore frame ptr til main scope
  movl  %ebp,%esp  #restore main start stak ptr
  popl  %ebp
  ret  #return fra main
#-------------------------
#end of program
-----------------------------------------------------
Slut på test af test_code_sdu_WhileDo.tony
:
::::::::::::::::::::::::::::::::::::::::::::::::::::::
:
```