Testprogramm für Kap.13 – Vektoren – Teil 2

```
/*
  Test program for chapter 13 "Vectors - Part 2"
*/
.section .bss
 .align 32.
 .lcomm dResult, 32
.section .data
  farrUnaligned: .float 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.0

  .align 32 # because of size of the array
  farrAligned: .float 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0

  .align 32
  farr1: .float 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8
  .align 32
  farr2: .float 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8

  .align 16
               #  -INF       (-)NaN      (+)NaN        +INF
  other: .long 0xff000000, 0xff600000, 0x7f600000, 0x7f000000

  .align 16
  darr1: .double 2.0, 3.0 # for square root

 .section .text
.globl main
.type main, @function
main:
  pushq %rbp

  # test p.3/4
  vmovups farrUnaligned, %ymm0  # ==> ymm0 = 8.0 7.0 6.0 5.0 4.0 3.0 2.0 1.0
  vextractf128 $0, %ymm0, %xmm0 # ==> xmm0 = 8.0 7.0 6.0 5.0

  vmovaps farrAligned, %ymm0    # ==> ymm0 = 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0
  vextractf128 $1, %ymm0, %xmm0 # ==> xmm0 = 5.0 6.0 7.0 8.0
  // testCommand for +4 ==> works!
  vmovss farrAligned+4, %xmm0   # ==> ymm0 = 2 0 0 0 0 0 0 0

  vmovaps farr1, %xmm1  # 1.1 1.2 1.3 1.4
  vmovaps farr2, %ymm2  # 2.1 2.2 2.3 2.4 2.5 2.5 2.7 2.8
  vinsertf128 $0, %xmm1, %ymm2, %ymm0 # ==> ymm0 = 1.1 ... 1.4 2.5 ... 2.8
  vinsertf128 $1, %xmm1, %ymm2, %ymm0 # ==> ymm0 = 2.1 ... 2.4 1.1 ... 1.4

  # test p.5
  vmovaps farr1, %xmm1
  vextractps $2, %xmm1, %ebx
  movl %ebx, %eax                    # ==> rax = 0x3fa66666 (float 1.3)
  # description for vinsertps see p.6
  vinsertps $53, farr2, %xmm1, %xmm0 # ==> xmm0 = 0.0 1.2 0.0 2.1

  # test p.8 - Addition
  vmovaps farr2, %ymm2
  # ymm2 + farr1
  vaddps farr1, %ymm2, %ymm0 # ==> ymm0 = 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6

  # test p.10 - Subtraction
  # ymm2 - farr1
  vsubps farr1, %ymm2, %ymm0 # ==> ymm0 = 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

  # test p.15 - Horizontal addition
```

```
vmovaps farr1, %ymm1
vhaddps farr2, %ymm1, %ymm0 # ==> ymm0 = 2.3 2.7 4.3 4.7 3.1 3.5 5.1 5.5

# test p.18 - Square root
vmovaps darr1, %xmm1
vsqrtpd %xmm1, %xmm0       # ==> xmm0 = 1.414 1.732

# test p.21 - float --> double
vmovaps farr1, %xmm1
vcvtps2pd %xmm1, %xmm0     # ==> xmm0 = (double) (1.1 1.2)

# test p.23 - double --> int (32 bit)
vmovaps darr1, %xmm1       # ==> xmm1 = 2.0 3.0
vcvtpd2dq %xmm1, %xmm0     # ==> xmm0 = 2   3
vmovaps %xmm0, dResult
leaq dResult, %rbx
movl (%rbx), %eax          # ==> eax = 2
movl 4(%rbx), %edx         # ==> edx = 3

# test p.28 - compare
vmovaps farr1, %xmm1
vmovaps farr2, %xmm2
vcmpps $1, %xmm2, %xmm1, %xmm0
       # xmm0 = 0xF...F 0xF...F 0xF...F 0xF...F

# test p.26 - other: -INF, (-)NaN, (+)naN, +INF

# case 1) ch. 12 Floating Point, p. 21
vmovss farr1, %xmm1  # xmm1 = 1.1 0.0 0.0 0.0
ucomiss other, %xmm1 # eflags: CF=0, ZF=0, PF=0 ==> 1.1 > -INF

# case 2) ch. 13, Vektoren Teil 2, S. 26
vmovaps farr1, %xmm1 # xmm = 1.1 1.2 1.3 1.4
       # $14 = greater
vcmpps $14, farr2, %xmm1, %xmm0
     # ==> xmm0 = 0x0...0 0x0...0 0x0...0 0x0...0

# case 3) ch. 13, Vektoren Teil 2, p. 26
vmovss farr1, %xmm1 # xmm = 1.1 0.0 0.0 0.0
vcmpunordss other, %xmm1, %xmm0 # no "packed" command
     # ==> xmm0 = 0x0...0

vmovss farr1, %xmm1 # xmm = 1.1 0.0 0.0 0.0
vcmpordss other+4, %xmm1, %xmm0 # no "packed" command
     # ==> xmm0 = 0xF...F

# exit main
movq $0, %rax
popq %rbp
ret
```