2 . Hausübung

Gruppe: - Ulf Gebhardt (rbg: hu56nifa)
- Michael Scholz (rbg: mi48azih)

Aufgabe 1: Matrikelprüfung:

```
.data
       # zu überprüfende Matrikelnummer einkommentieren:

       # matr: .long 0,9,3,9,1,8,2 # --> falsch
       # matr: .long 1,2,3,4,5,6,7 # --> falsch
         matr: .long 0,6,1,1,8,1,3 # --> richtig
       # matr: .long 1,3,9,8,1,4,0 # --> falsch
       # matr: .long 1,5,6,9,2,0,3 # --> falsch


intout:
.string "1 -> Matrikelnummer richtig\n2 -> Matrikelnummer falsch\nWert: %d\n"

.text

.globl main


main:

       xor %ecx, %ecx              # %ecx = 0
       xor %edi, %edi              # 0 in Indexregister ("xor" ist schneller als "movl $0,
                                   # reg")


       movl matr(,%edi,4), %ebx    # liest erste Stelle (a)
       imull $9, %ebx              # a * 9
       addl %ebx, %ecx             # %ecx += a

       inc %edi                    # %edi += 1
       movl matr(,%edi,4), %ebx    # liest zweite Stelle (b)
       imull $7, %ebx              # b * 7
       addl %ebx, %ecx             # %ecx += b

       inc %edi                    # %edi += 1
       movl matr(,%edi,4), %ebx    # liest dritte Stelle (c)
       imull $3, %ebx              # c * 3
       addl %ebx, %ecx             # %ecx += c

       inc %edi                    # %edi += 1
       movl matr(,%edi,4), %ebx    # liest vierte Stelle (d)
       imull $9, %ebx              # d * 9
       addl %ebx, %ecx             # %ecx += c

       inc %edi                    # %edi += 1
       movl matr(,%edi,4), %ebx    # liest fünfte Stelle (e)
       imull $7, %ebx              # e * 7
       addl %ebx, %ecx             # %ecx += c

       inc %edi                    # %edi += 1
       movl matr(,%edi,4), %ebx    # liest sechste Stelle (f)
       imull $3, %ebx              # f * 3
       addl %ebx, %ecx             # %ecx += c
```

```
        movl %ecx, %eax             # bereite für Division vor

        movl $10, %ebx
        xor %edx, %edx              # %edx leeren, sonst loating point exception, da noch
                                    # alter Wert in %edx steht

        divl %ebx                   # %eax / %ebx (10) Quotient liegt nun in %eax, Rest in
                                    # %edx

        inc %edi                    # %edi += 1
        movl matr(,%edi,4), %eax    # %eax = siebte Stelle (g)



        cmp %edx, %eax              # Vergleiche Rest mit g
        je .true                    # rest = g

        jmp .false                  # rest != g

        .true:
        movl $1, %eax               # Ausgabe true -> 1
        jmp .ende                   # "false-Label" überspringen

        .false:
        movl $0, %eax               # Ausgabe false -> 0

        .ende:


# Wert im %eax ausgeben
pushl %eax
pushl $intout
call printf


# Exit
movl $1, %eax
int $0x80
```

Aufgabe 2: Sortieren:

```
.data

        # zu sortierende List einkommentieren
          list: .long 10,9,8,7,6,5,4,3,2,1
        # list: .long 16,5,18,12,11,66,62,1,9,33

        # Länge der Liste:
        length: .long 10

intout:

.string "Sortierte Liste: %d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n"

.text

.globl main

main:

        movl length, %ecx    # %ecx = n
        subl $1, %ecx        # n - 1


        movl $0, %edi        # %edi mit 0 initialisieren
```

```
#############################################################################

.schleife_aussen:

pushl %ecx    # Speicher den Wert von %ecx auf Stack

       ###########################################################

       .schleife_innen:


              movl list(,%edi, 4), %eax  # %eax = j
              inc %edi                   # %edi um 1 erhöhen
              movl list(,%edi, 4), %ebx  # %ebx = j+1
              dec %edi                   # %edi wieder dekrementieren

              cmp %eax, %ebx             # vergleiche nun j und j+1 und springe zu den
                                         # passenden Labels
              jb .true

              jmp .false

                     .true:
                            # Registertausch:
                            movl %ebx, list(,%edi, 4) # zurück ins Datenfeld schreiben
                            inc %edi
                            movl %eax, list(,%edi, 4) # zurück ins Datenfeld schreiben
                            dec %edi

                     .false:
                            # Weiter im Code

              inc %edi      # %edi um 1 erhöhen, um nun das nächste Paar im Array zu
                            # vergleichen


       loop .schleife_innen

       ###########################################################


popl %ecx           # ursprünglichen Wert von %ecx vom Stack holen

xor %edi, %edi      # %edi wieder auf 0 setzen

loop .schleife_aussen

#############################################################################

#### Ausgabe ####

movl length, %esi          # Kopiere Arraygröße nach %esi(= k)

.printElement:

       cmpl $0 , %esi      # Vergleiche %esi = 0 ?
       je .callPrintf      # Wenn ja ,springe zu callPrintf
       dec %esi            # Veringere %esi um 1

       movl list(,%esi,4) , %ecx  # Kopiere a[k] nach %ecx
       pushl %ecx                 # pushl auf Stack
       jmp .printElement

.callPrintf:
pushl $intout
call  printf

# Exit
movl $1, %eax
int $0x80
```