```
;*******************************************************************
;* htw saar - Fakultaet fuer Ingenieurwissenschaften               *
;* Labor fuer Eingebettete Systeme                                 *
;* Mikroprozessortechnik                                           *
;*******************************************************************
;* Assembler_Startup.S:                                            *
;* Programmrumpf fuer Assembler-Programme mit dem Keil             *
;* Entwicklungsprogramm uVision fuer ARM-Mikrocontroller           *
;*******************************************************************
;* Aufgabe-Nr.:            *              1                    *
;*                         *                                   *
;*******************************************************************
;* Gruppen-Nr.:            *              5                    *
;*                         *                                   *
;*******************************************************************
;* Name / Matrikel-Nr.:    *              Valentin Straßer    5014379                  *
;*                                        Michal Roziel       5012845          *
;*                                 *                                  *
;*******************************************************************
;* Abgabedatum:            *               19.12.2024                 *
;*                                 *                                  *
;*******************************************************************
;*******************************************************************
;* Daten-Bereich bzw. Daten-Speicher                               *
;*******************************************************************
                        AREA                Daten, DATA, READWRITE
Datenanfang
STR_1                   EQU                 Datenanfang  + 0x100
Stack_Anfang            EQU                 Datenanfang  + 0x200
Top_Stack               EQU                 Stack_Anfang + 0x400
STR_2                   EQU                 Top_Stack
;*******************************************************************
;* Programm-Bereich bzw. Programm-Speicher                         *
;*******************************************************************
                        AREA                Programm, CODE, READONLY
                        ARM
Reset_Handler           MSR                 CPSR_c, #0x10
;*******************************************************************
;* Hier das eigene (Haupt-)Programm einfuegen                      *
;*******************************************************************
            LDR         SP,=Top_Stack       ; Adresse des Werts laden
            LDR         R0,=STR_1           ; Wert laden
            LDR         R9,=0x0000FFFF      ; Zur Sicherheit : Begrenzung auf 16 Bits
            BL          atouI
            AND         R0,R0,R9
            BL          berechnung
            AND         R0,R0,R9            ; Zur Absicherung : Begrenzung auf 16 Bits
            LDR         R1,=STR_2
            BL          uitoa
;*******************************************************************
;* Ende des eigenen (Haupt-)Programms                              *
;*******************************************************************
endlos      B           endlos
;*******************************************************************
;* ab hier Unterprogramme                                          *
;*******************************************************************
;*******************************************************************************************************

; Aufgabe 2.1 ATOUI - ASCII ZU UNSIGNED INTEGER
; Eingabe : R0, Adresse des ersten Zeichens des Strings - STR_1 bei 0x40000000
; Ausgabe : R0 - konvertiert in 32 Bit Unsigned Integer Zahl
;*******************************************************************************************************
atouI
            STMFD       SP!, {R1-R4, R14}   ; Speichere Register R1-R4 und Rücksprungadresse auf dem
Stack
            MOV         R2, #10             ; R2 = 10 (Basis für Dezimalberechnung)
            MOV         R3, #0              ; R3 = 0 (Startwert für die Berechnung)
schleife_atouI
            LDRB        R1, [R0], #1        ; Lade aktuelles Zeichen aus der Speicheradresse R0, R0++
            MOV         R4, R3              ; Speichere bisherigen Wert von R3 in R4
            CMP         R1, #0x00           ; Prüfe, ob Ende der Zeichenkette erreicht (Null-Terminator)
            SUBNE       R1, R1, #0x30       ; Falls nicht, konvertiere ASCII-Zeichen in Dezimalwert
```

```
71              MLANE       R3, R4, R2, R1       ; Multipliziere bisherigen Wert mit 10 und addiere neuen
    Wert
72              BNE         schleife_atouI       ; Wiederhole, falls noch Zeichen übrig
73
74              MOV         R0, R4               ; Speichere Endergebnis in R0
75              LDMFD       SP!, {R1-R4, R14}    ; Wiederherstellen der ursprünglichen Registerwerte
76              BX          LR                   ; Rückkehr zur aufrufenden Funktion
77  ;*************************************************************************************************
78  ; Aufgabe 4.2 Berechne Y = ((2/5) X) ZUM QUADRAT
79  ; Eingabe : R0, 16 Bit Signed ganze Zahl
80  ; Ausgabe : R0 - Funktionswert Y
81  ;*************************************************************************************************
82  berechnung
83              STMFD       SP!, {R1-R4, LR}     ; Speichere Register R1-R4 und Rücksprungadresse
84              MOV         R0, R0, LSL #16      ; Verschiebe die unteren 16 Bits nach oben
85              ASR         R0, R0, #16          ; Verschiebe arithmetisch nach rechts (übernimmt Vorzeichen)
86              CMP         R0, #0               ; Prüfe, ob die Zahl negativ ist
87              RSBMI       R0, R0, #0           ; Falls negativ, mache sie positiv
88
89              MOV         R1, R0               ; R1 = Eingabewert (X)
90              MOV         R2, R1, LSL #1       ; R2 = 2 * X (Linksschieben um 1 Bit)
91
92              LDR         R1, =0xCCCCCCCD      ; Lade Magic Number für Division durch 10 -> Skript
93              UMULL       R3, R4, R2, R1       ; R2 * Magic Number -> Ergebnis in R3:R4
94              MOV         R2, R4, LSR #2       ; Teile Ergebnis (R4) durch 4 (Rechtsverschiebung um 2 Bits)
95
96              MUL         R0, R2, R2           ; Quadriere das Ergebnis (R0 = R2 * R2)
97              BX          LR                   ; Rückkehr zur aufrufenden Funktion
98              LDMFD       SP!, {R1-R4, LR}     ; Wiederherstellen der ursprünglichen Registerwerte
99  ;*************************************************************************************************
100 ; Aufgabe 4.3 UNSIGNED INTEGER ZU ASCII
101 ; Eingabe : R0, 16 Bit Unsigned ganze Zahl , R1-Adresse für 1. Zeichen des zu erzeugenden Strings-STR_2
102 ; Ausgabe : ASCII konvertierte Zahl an der Adresse von STR_2
103 ;*************************************************************************************************
104 uitoa
105             STMFD       SP!, {R2-R7, LR}     ; Speichere Register R2-R7 und Rücksprungadresse
106             MOV         R2, #10              ; Basis (10) für Division
107             LDR         R3, =0xCCCD          ; Magic Number für Division durch 10
108             MOV         R4, #0               ; Zähler für die Anzahl der Ziffern
109
110 schleife_uitoa
111             CMP         R0, #0               ; Prüfe, ob der Wert 0 ist
112             MULNE       R6, R0, R3           ; Multipliziere mit der Magic Number
113             MOVNE       R5, R6, LSR #19      ; Quotient ausrechnen (Rechtsverschiebung um 19 Bit)
114             MULNE       R6, R5, R2           ; R6 = R5 * 10
115             SUBNE       R6, R0, R6           ; Rest berechnen (R6 = R0 - (R5 * 10))
116             ADDNE       R6, #0x30            ; Umwandeln in ASCII ('0'-'9')
117             STMFDNE     SP!, {R6}            ; Speichere das Zeichen auf dem Stack
118             MOVNE       R0, R5               ; Aktualisiere R0 für nächste Iteration
119             ADDNE       R4, R4, #1           ; Zähler erhöhen
120             BNE         schleife_uitoa            ; Wiederhole, falls noch Ziffern übrig
121 revstr
122             CMP         R4, #0               ; Prüfe, ob noch Zeichen vorhanden sind
123             LDMFDNE     SP!, {R6}            ; Hole Zeichen vom Stack
124             STRBNE      R6, [R1], #1         ; Schreibe Zeichen in Speicher, R1++
125             SUBNE       R4, R4, #1           ; Zähler verringern
126             BNE         revstr               ; Wiederhole, bis alle Zeichen verarbeitet sind
127
128             MOV         R3, #0x00            ; Null-Terminator
129             STRB        R3, [R1]             ; Schreibe Null-Terminator ans Ende
130
131             LDMFD       SP!, {R2-R7, LR}     ; Wiederherstellen der ursprünglichen Registerwerte
132             BX          LR                   ; Rückkehr zur aufrufenden Funktion
133 ;*****************************************************************
134 ;* Konstanten im CODE-Bereich                                    *
135 ;*****************************************************************
136 String_1                  DCB                 "65535",0x00
137 ;*****************************************************************
138 ;* Ende der Programm-Quelle                                      *
139 ;*****************************************************************
140             ALIGN
141             END
```