Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
        mov     r0, #4
        mov     r2, #2
        mov     r3, #3                  @ in r0 steht dann 0x
        str     r2, [r0]
        str     r3, [r0, #4]
        ldr     r1, [r0], #4            @ in r0 steht dann 0x
                                        @ in r1 steht dann 0x
        mov     r0, #4
        str     r1, [r0, #4]            @ in r0 steht dann 0x
        mov     r0, #4
        ldr     r1, [r0]!               @ in r0 steht dann 0x
        mov     r0, #2
        strb    r1, [r0, #6]!           @ in r0 steht dann 0x
```

Welcher Wert steht an der Adresse 0x4:

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
        mov     r0, #4
        mov     r2, #2
        mov     r3, #3              @ in r0 steht dann 0x4
        str     r2, [r0]
        str     r3, [r0, #4]
        ldr     r1, [r0], #4        @ in r0 steht dann 0x
                                    @ in r1 steht dann 0x
        mov     r0, #4
        str     r1, [r0, #4]        @ in r0 steht dann 0x
        mov     r0, #4
        ldr     r1, [r0]!           @ in r0 steht dann 0x
        mov     r0, #2
        strb    r1, [r0, #6]!       @ in r0 steht dann 0x
```

Welcher Wert steht an der Adresse 0x4:

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
mov     r0, #4
mov     r2, #2
mov     r3, #3              @ in r0 steht dann 0x4
str     r2, [r0]
str     r3, [r0, #4]
ldr     r1, [r0], #4        @ in r0 steht dann 0x8
                            @ in r1 steht dann 0x
mov     r0, #4
str     r1, [r0, #4]        @ in r0 steht dann 0x
mov     r0, #4
ldr     r1, [r0]!           @ in r0 steht dann 0x
mov     r0, #2
strb    r1, [r0, #6]!       @ in r0 steht dann 0x
```

Welcher Wert steht an der Adresse 0x4:

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
mov     r0, #4
mov     r2, #2
mov     r3, #3              @ in r0 steht dann 0x4
str     r2, [r0]
str     r3, [r0, #4]
ldr     r1, [r0], #4        @ in r0 steht dann 0x8
                            @ in r1 steht dann 0x2
mov     r0, #4
str     r1, [r0, #4]        @ in r0 steht dann 0x
mov     r0, #4
ldr     r1, [r0]!           @ in r0 steht dann 0x
mov     r0, #2
strb    r1, [r0, #6]!       @ in r0 steht dann 0x
```

Welcher Wert steht an der Adresse 0x4:

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
mov     r0, #4
mov     r2, #2
mov     r3, #3              @ in r0 steht dann 0x4
str     r2, [r0]
str     r3, [r0, #4]
ldr     r1, [r0], #4        @ in r0 steht dann 0x8
                            @ in r1 steht dann 0x2
mov     r0, #4
str     r1, [r0, #4]        @ in r0 steht dann 0x4
mov     r0, #4
ldr     r1, [r0]!           @ in r0 steht dann 0x
mov     r0, #2
strb    r1, [r0, #6]!       @ in r0 steht dann 0x
```

Welcher Wert steht an der Adresse 0x4:

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
mov     r0, #4
mov     r2, #2
mov     r3, #3              @ in r0 steht dann 0x4
str     r2, [r0]
str     r3, [r0, #4]
ldr     r1, [r0], #4        @ in r0 steht dann 0x8
                            @ in r1 steht dann 0x2
mov     r0, #4
str     r1, [r0, #4]        @ in r0 steht dann 0x4
mov     r0, #4
ldr     r1, [r0]!           @ in r0 steht dann 0x4
mov     r0, #2
strb    r1, [r0, #6]!       @ in r0 steht dann 0x
```

Welcher Wert steht an der Adresse 0x4:

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
mov     r0, #4
mov     r2, #2
mov     r3, #3              @ in r0 steht dann 0x4
str     r2, [r0]
str     r3, [r0, #4]
ldr     r1, [r0], #4        @ in r0 steht dann 0x8
                            @ in r1 steht dann 0x2
mov     r0, #4
str     r1, [r0, #4]        @ in r0 steht dann 0x4
mov     r0, #4
ldr     r1, [r0]!           @ in r0 steht dann 0x4
mov     r0, #2
strb    r1, [r0, #6]!       @ in r0 steht dann 0x8
```

Welcher Wert steht an der Adresse 0x4:

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
mov     r0, #4
mov     r2, #2
mov     r3, #3              @ in r0 steht dann 0x4
str     r2, [r0]
str     r3, [r0, #4]
ldr     r1, [r0], #4        @ in r0 steht dann 0x8
                            @ in r1 steht dann 0x2
mov     r0, #4
str     r1, [r0, #4]        @ in r0 steht dann 0x4
mov     r0, #4
ldr     r1, [r0]!           @ in r0 steht dann 0x4
mov     r0, #2
strb    r1, [r0, #6]!       @ in r0 steht dann 0x8
```

Welcher Wert steht an der Adresse 0x4: **2**

Welcher Wert steht an der Adresse 0x8:

Aufgabe 1: (10 Punkte)

Ergänzen Sie die Kommentarzeilen

```
mov   r0, #4
mov   r2, #2
mov   r3, #3          @ in r0 steht dann 0x4
str   r2, [r0]
str   r3, [r0, #4]
ldr   r1, [r0], #4    @ in r0 steht dann 0x8
                      @ in r1 steht dann 0x2
mov   r0, #4
str   r1, [r0, #4]    @ in r0 steht dann 0x4
mov   r0, #4
ldr   r1, [r0]!       @ in r0 steht dann 0x4
mov   r0, #2
strb  r1, [r0, #6]!   @ in r0 steht dann 0x8
```

Welcher Wert steht an der Adresse 0x4: **2**

Welcher Wert steht an der Adresse 0x8: **2**

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov     r0, #254            @ in r0 steht dann 0x
and     r0, r0, #126        @ in r0 steht dann 0x
mov     r1, #255            @ in r1 steht dann 0x
eor     r0, r0, r1          @ in r0 steht dann 0x
mov     r0, #0x81
bic     r0, r0, #20         @ in r0 steht dann 0x
eor     r0, r0, r0          @ in r0 steht dann 0x
mvn     r0, #1              @ in r0 steht dann 0x
mov     r0, #0x81
orr     r0, r0, 0x82        @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov   r0, #254         @ in r0 steht dann 0xfe
and   r0, r0, #126     @ in r0 steht dann 0x
mov   r1, #255         @ in r1 steht dann 0x
eor   r0, r0, r1       @ in r0 steht dann 0x
mov   r0, #0x81
bic   r0, r0, #20      @ in r0 steht dann 0x
eor   r0, r0, r0       @ in r0 steht dann 0x
mvn   r0, #1           @ in r0 steht dann 0x
mov   r0, #0x81
orr   r0, r0, 0x82     @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov     r0, #254          @ in r0 steht dann 0xfe
and     r0, r0, #126      @ in r0 steht dann 0x7e
mov     r1, #255          @ in r1 steht dann 0x
eor     r0, r0, r1        @ in r0 steht dann 0x
mov     r0, #0x81
bic     r0, r0, #20       @ in r0 steht dann 0x
eor     r0, r0, r0        @ in r0 steht dann 0x
mvn     r0, #1            @ in r0 steht dann 0x
mov     r0, #0x81
orr     r0, r0, 0x82      @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov     r0, #254          @ in r0 steht dann 0xfe
and     r0, r0, #126      @ in r0 steht dann 0x7e
mov     r1, #255          @ in r1 steht dann 0xff
eor     r0, r0, r1        @ in r0 steht dann 0x
mov     r0, #0x81
bic     r0, r0, #20       @ in r0 steht dann 0x
eor     r0, r0, r0        @ in r0 steht dann 0x
mvn     r0, #1            @ in r0 steht dann 0x
mov     r0, #0x81
orr     r0, r0, 0x82      @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov   r0, #254          @ in r0 steht dann 0xfe
and   r0, r0, #126      @ in r0 steht dann 0x7e
mov   r1, #255          @ in r1 steht dann 0xff
eor   r0, r0, r1        @ in r0 steht dann 0x81
mov   r0, #0x81
bic   r0, r0, #20       @ in r0 steht dann 0x
eor   r0, r0, r0        @ in r0 steht dann 0x
mvn   r0, #1            @ in r0 steht dann 0x
mov   r0, #0x81
orr   r0, r0, 0x82      @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov     r0, #254          @ in r0 steht dann 0xfe
and     r0, r0, #126      @ in r0 steht dann 0x7e
mov     r1, #255          @ in r1 steht dann 0xff
eor     r0, r0, r1        @ in r0 steht dann 0x81
mov     r0, #0x81
bic     r0, r0, #20       @ in r0 steht dann 0x81
eor     r0, r0, r0        @ in r0 steht dann 0x
mvn     r0, #1            @ in r0 steht dann 0x
mov     r0, #0x81
orr     r0, r0, 0x82      @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov     r0, #254            @ in r0 steht dann 0xfe
and     r0, r0, #126        @ in r0 steht dann 0x7e
mov     r1, #255            @ in r1 steht dann 0xff
eor     r0, r0, r1          @ in r0 steht dann 0x81
mov     r0, #0x81
bic     r0, r0, #20         @ in r0 steht dann 0x81
eor     r0, r0, r0          @ in r0 steht dann 0x0
mvn     r0, #1              @ in r0 steht dann 0x
mov     r0, #0x81
orr     r0, r0, 0x82        @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov   r0, #254         @ in r0 steht dann 0xfe
and   r0, r0, #126     @ in r0 steht dann 0x7e
mov   r1, #255         @ in r1 steht dann 0xff
eor   r0, r0, r1       @ in r0 steht dann 0x81
mov   r0, #0x81
bic   r0, r0, #20      @ in r0 steht dann 0x81
eor   r0, r0, r0       @ in r0 steht dann 0x0
mvn   r0, #1           @ in r0 steht dann 0xfffffffe
mov   r0, #0x81
orr   r0, r0, #0x82    @ in r0 steht dann 0x
```

Aufgabe 2: (8 Punkte)

Was steht im jeweiligen Register nach Ausführung des Befehles ? Geben Sie die Werte hexadezimal an.

```
mov     r0, #254            @ in r0 steht dann 0xfe
and     r0, r0, #126        @ in r0 steht dann 0x7e
mov     r1, #255            @ in r1 steht dann 0xff
eor     r0, r0, r1          @ in r0 steht dann 0x81
mov     r0, #0x81
bic     r0, r0, #20         @ in r0 steht dann 0x81
eor     r0, r0, r0          @ in r0 steht dann 0x0
mvn     r0, #1              @ in r0 steht dann 0xfffffffe
mov     r0, #0x81
orr     r0, r0, #0x82       @ in r0 steht dann 0x83
```

Aufgabe 3 (12 Punkte)

Folgende C-Funktion strcmp (String Compare) vergleicht zwei Zeichenkette und gibt im Fall der Gleichheit den Wert 0 zurück bzw. im Fall der Ungleichheit die Differenz der ersten ungleichen Zeichen in der Zeichenkette.

```
int strcmp(const char *s1, const char *s2)
{
  unsigned int  u1, u2;

  while (1)
    {
      u1 = (unsigned int) *s1++;
      u2 = (unsigned int) *s2++;
      if (u1 != u2)
                         return u1 - u2;
      if (u1 == '\0')
                         return 0;
    }
}
```

Diese Funktion wird nun in ARM Assembler dargestellt. Hierbei haben sich Fehler eingeschlichen. Verbessern Sie das Assembler Programm, ohne die Befehlsreihenfolge zu ändern.

```
strcmp:
        mov     r2, r0
.L7:
        ldr     r0, [r2], #4
        ldr     r3, [r1], #4
        cmp     r0, r3
        subeq   r0, r3, r0
        moveq   lr, pc
        cmp     r0, #0
        bne     .L7
        mov     lr, pc
```

```
strcmp:
        mov     r2, r0
.L7:
        ldrb    r0, [r2], #1
        ldr     r3, [r1], #4
        cmp     r0, r3
        subeq   r0, r3, r0
        moveq   lr, pc
        cmp     r0, #0
        bne     .L7
        mov     lr, pc
```

```
strcmp:
        mov     r2, r0
.L7:
        ldrb    r0, [r2], #1
        ldrb    r3, [r1], #1
        cmp     r0, r3
        subeq   r0, r3, r0
        moveq   lr, pc
        cmp     r0, #0
        bne     .L7
        mov     lr, pc
```

```
strcmp:
        mov     r2, r0
.L7:
        ldrb    r0, [r2], #1
        ldrb    r3, [r1], #1
        cmp     r0, r3
        rsbne   r0, r3, r0
        movne   pc,lr
        cmp     r0, #0
        bne     .L7
        mov     pc,lr
```

In welchen Registern stehen die Parameter der Funktion beim Eintritt in die Funktion strcmp ?

In welchem Register steht der Rückgabewert der Funktion beim Austritt aus der Funktion strcmp ?

In welchen Registern stehen die Parameter der Funktion beim Eintritt in die Funktion strcmp ?

R0, R1

In welchem Register steht der Rückgabewert der Funktion beim Austritt aus der Funktion strcmp ?

In welchen Registern stehen die Parameter der Funktion beim Eintritt in die Funktion strcmp ?

R0, R1

In welchem Register steht der Rückgabewert der Funktion beim Austritt aus der Funktion strcmp ?

R0

Aufgabe 4 (8 Punkte)

Welchen Wert hat das Register R0 nach diesem kleinen Programm ?

```
LDR     r0, =0x33446677
EOR     ip , r0 , r0 , ror #16
BIC     ip , ip , #0xff0000
MOV     r0 , r0 , ror #8
EOR     r0 , r0 , ip , lsr #8
```

Füllen Sie hierzu folgende Tabelle aus:

Befehl		Inhalt von R0	Inhalt von IP
LDR	r0, =0x33446677	?	-
EOR	ip , r0 , r0 , ror #16	-	?
BIC	ip , ip , #0xff0000	-	?
MOV	r0 , r0 , ror #8	?	-
EOR	r0 , r0 , ip , lsr #8	?	?

Aufgabe 4 (8 Punkte)

Welchen Wert hat das Register R0 nach diesem kleinen Programm ?

```
LDR     r0, =0x33446677
EOR     ip , r0 , r0 , ror #16
BIC     ip , ip , #0xff0000
MOV     r0 , r0 , ror #8
EOR     r0 , r0 , ip , lsr #8
```

Füllen Sie hierzu folgende Tabelle aus:

Befehl		Inhalt von R0	Inhalt von IP
LDR	r0, =0x33446677	0x33446677	-
EOR	ip , r0 , r0 , ror #16	-	?
BIC	ip , ip , #0xff0000	-	?
MOV	r0 , r0 , ror #8	?	-
EOR	r0 , r0 , ip , lsr #8	?	?

Aufgabe 4 (8 Punkte)

Welchen Wert hat das Register R0 nach diesem kleinen Programm ?

```
LDR     r0, =0x33446677
EOR     ip , r0 , r0 , ror #16
BIC     ip , ip , #0xff0000
MOV     r0 , r0 , ror #8
EOR     r0 , r0 , ip , lsr #8
```

Füllen Sie hierzu folgende Tabelle aus:

Befehl		**Inhalt von R0**	**Inhalt von IP**
LDR	r0, =0x33446677	0x33446677	-
EOR	ip , r0 , r0 , ror #16	-	0x55335533
BIC	ip , ip , #0xff0000	-	?
MOV	r0 , r0 , ror #8	?	-
EOR	r0 , r0 , ip , lsr #8	?	?

Aufgabe 4 (8 Punkte)

Welchen Wert hat das Register R0 nach diesem kleinen Programm ?

```
LDR     r0, =0x33446677
EOR     ip , r0 , r0 , ror #16
BIC     ip , ip , #0xff0000
MOV     r0 , r0 , ror #8
EOR     r0 , r0 , ip , lsr #8
```

Füllen Sie hierzu folgende Tabelle aus:

Befehl		Inhalt von R0	Inhalt von IP
LDR	r0, =0x33446677	0x33446677	-
EOR	ip , r0 , r0 , ror #16	-	0x55335533
BIC	ip , ip , #0xff0000	-	0x55005533
MOV	r0 , r0 , ror #8	?	-
EOR	r0 , r0 , ip , lsr #8	?	?

Aufgabe 4 (8 Punkte)

Welchen Wert hat das Register R0 nach diesem kleinen Programm ?

```
LDR     r0, =0x33446677
EOR     ip , r0 , r0 , ror #16
BIC     ip , ip , #0xff0000
MOV     r0 , r0 , ror #8
EOR     r0 , r0 , ip , lsr #8
```

Füllen Sie hierzu folgende Tabelle aus:

Befehl		**Inhalt von R0**	**Inhalt von IP**
LDR	r0, =0x33446677	0x33446677	-
EOR	ip , r0 , r0 , ror #16	-	0x55335533
BIC	ip , ip , #0xff0000	-	0x55005533
MOV	r0 , r0 , ror #8	0x77334466	-
EOR	r0 , r0 , ip , lsr #8	?	?

Aufgabe 4 (8 Punkte)

Welchen Wert hat das Register R0 nach diesem kleinen Programm ?

```
LDR   r0, =0x33446677
EOR   ip , r0 , r0 , ror #16
BIC   ip , ip , #0xff0000
MOV   r0 , r0 , ror #8
EOR   r0 , r0 , ip , lsr #8
```

Füllen Sie hierzu folgende Tabelle aus:

Befehl		**Inhalt von R0**	**Inhalt von IP**
LDR	r0, =0x33446677	0x33446677	-
EOR	ip , r0 , r0 , ror #16	-	0x55335533
BIC	ip , ip , #0xff0000	-	0x55005533
MOV	r0 , r0 , ror #8	0x77334466	-
EOR	r0 , r0 , ip , lsr #8	0x77664433	?

Aufgabe 4 (8 Punkte)

Welchen Wert hat das Register R0 nach diesem kleinen Programm ?

```
LDR     r0, =0x33446677
EOR     ip , r0 , r0 , ror #16
BIC     ip , ip , #0xff0000
MOV     r0 , r0 , ror #8
EOR     r0 , r0 , ip , lsr #8
```

Füllen Sie hierzu folgende Tabelle aus:

Befehl		**Inhalt von R0**	**Inhalt von IP**
LDR	r0, =0x33446677	0x33446677	-
EOR	ip , r0 , r0 , ror #16	-	0x55335533
BIC	ip , ip , #0xff0000	-	0x55005533
MOV	r0 , r0 , ror #8	0x77334466	-
EOR	r0 , r0 , ip , lsr #8	0x77664433	(0x00550055)

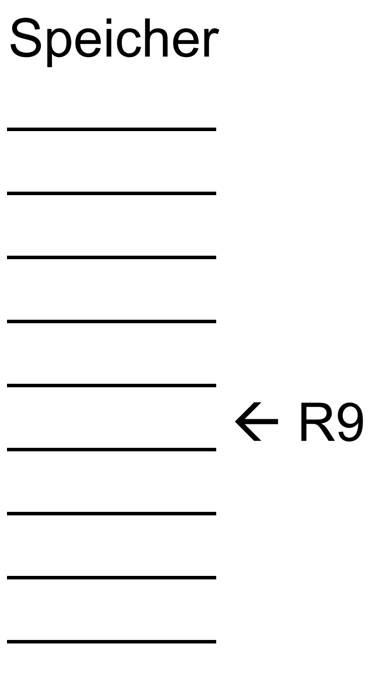
Was leistet das Programm?

Was leistet das Programm?

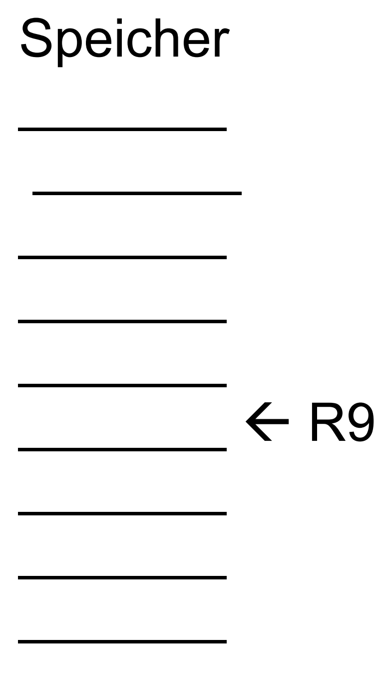
Das Programm kehrt die Bytereihenfolge in einem 32 Bit Wort um.

Aufgabe 5 (8 Punkte)

Ausgangssituation: In R9 ist eine Adresse gespeichert, die gemäß der folgenden Abbildungen auf den Speicher zeigt. Wie sieht der Speicher aus nach Ausführung des jeweiligen Blocktransferbefehls. Ergänzen Sie hierzu die Speichertabelle und setzen Sie einen Pfeil mit der Angabe R9' an die Speicheradresse, auf die das Register R9 nach Ausführung des Befehls zeigt.



STMIA R9!, {R0, R2, R6}



STMIB R9, {R0, R2, R6}

Aufgabe 5 (8 Punkte)

Ausgangssituation: In R9 ist eine Adresse gespeichert, die gemäß der folgenden Abbildungen auf den Speicher zeigt. Wie sieht der Speicher aus nach Ausführung des jeweiligen Blocktransferbefehls. Ergänzen Sie hierzu die Speichertabelle und setzen Sie einen Pfeil mit der Angabe R9’ an die Speicheradresse, auf die das Register R9 nach Ausführung des Befehls zeigt.

Speicher		Speicher	
	← **R9**	R6	
R6		R2	
R2		R0	
R0	← R9		← R9
STMIA R9!, {R0, R2, R6}		STMIB R9, {R0, R2, R6}	

Aufgabe 5 (8 Punkte)

Ausgangssituation: In R9 ist eine Adresse gespeichert, die gemäß der folgenden Abbildungen auf den Speicher zeigt. Wie sieht der Speicher aus nach Ausführung des jeweiligen Blocktransferbefehls. Ergänzen Sie hierzu die Speichertabelle und setzen Sie einen Pfeil mit der Angabe R9' an die Speicheradresse, auf die das Register R9 nach Ausführung des Befehls zeigt.

Speicher	
R6	← R9
R2	
R0	
	← **R9**

STMDA R9!, {R0, R2, R6}

Speicher	
	← R9
R6	
R2	
R0	

STMDB R9, {R0, R2, R6} }

Schauen Sie sich die 3 Fenster an und beantworten Sie hierzu folgende Fragen.

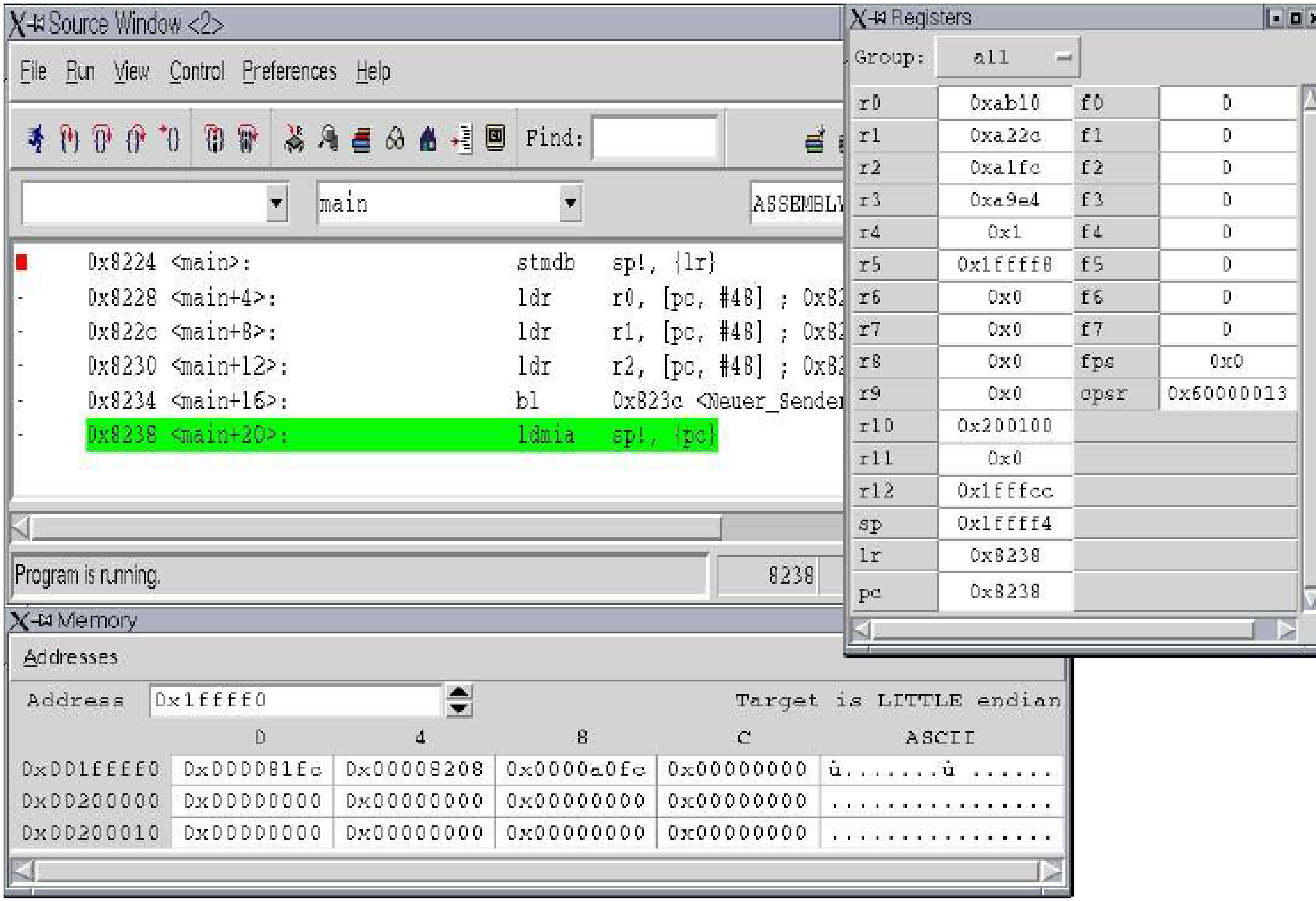
1. Welche Adresse steht nach Ausführung des gezeigten Befehl im Register PC?
(2 Punkte)
2. Welche Adresse steht nach Ausführung des gezeigten Befehl im Register SP?
(1 Punkt)
2. Geben Sie an ob die genannten Flags 1 oder 0 sind
(4 Punkte)

Carryflag = ___

Zeroflag = ___

Negativflag = ___

Overflowflag = ___


Source Window <2>
File Run View Control Preferences Help
Find:
main
ASSEMBLY
0x8224 <main>: stmdb sp!, {lr}
0x8228 <main+4>: ldr r0, [pc, #48] ; 0x8
0x822c <main+8>: ldr r1, [pc, #48] ; 0x8
0x8230 <main+12>: ldr r2, [pc, #48] ; 0x8
0x8234 <main+16>: bl 0x823c <Neuer_Sender
0x8238 <main+20>: ldmia sp!, {pc}
Program is running.
8238
Registers
Group: all
r0 0xab10 f0 0
r1 0xa22c f1 0
r2 0xa1fc f2 0
r3 0xa9e4 f3 0
r4 0x1 f4 0
r5 0x1ffff8 f5 0
r6 0x0 f6 0
r7 0x0 f7 0
r8 0x0 fps 0x0
r9 0x0 cpsr 0x60000013
r10 0x200100
r11 0x0
r12 0x1fffcc
sp 0x1ffff4
lr 0x8238
pc 0x8238
Memory
Addresses
Address 0x1ffff0
Target is LITTLE endian
0 4 8 C ASCII
0x001ffff0 0x000081fc 0x00008208 0x0000a0fc 0x00000000 ü.......ü ......
0x00200000 0x00000000 0x00000000 0x00000000 0x00000000 ....
0x00200010 0x00000000 0x00000000 0x00000000 0x00000000 ....

12 Punkte

a)

Welche Werte haben die Register R0-R3, R13 jeweils nach Ausführung der folgenden Befehle.

Tragen Sie die Lösung in die Tabelle jeweils in den Zeilen mit dem stm und ldm Befehl ein.

Geben Sie die Register R0 – R4 dezimal und das Register R9 hexadezimal an.

Befehl	R0	R1	R2	R3	R13
Initialisierung	0	1	2	3	0x200
stmda r13!, {r0-r2}					
ldmib r13, {r2,r3}					
Initialisierung	0	1	2	3	0x200
stmdb r13!, {r0-r3}					
ldmib r13!, {r0-r2}					

12 Punkte

a)

Welche Werte haben die Register R0-R3, R13 jeweils nach Ausführung der folgenden Befehle.

Tragen Sie die Lösung in die Tabelle jeweils in den Zeilen mit dem stm und ldm Befehl ein.

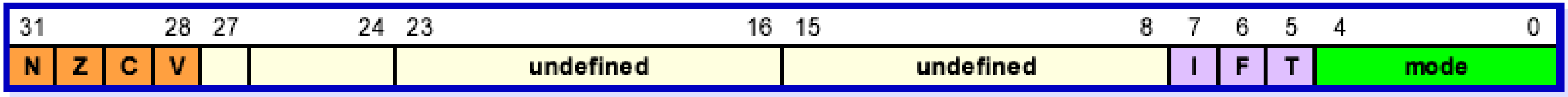
Geben Sie die Register R0 – R4 dezimal und das Register R9 hexadezimal an.

Befehl	R0	R1	R2	R3	R13
Initialisierung	0	1	2	3	0x200
stmda r13!, {r0-r2}					
ldmib r13, {r2,r3}					
Initialisierung	0	1	2	3	0x200
stmdb r13!, {r0-r3}					
ldmib r13!, {r0-r2}					

b)
Welche Stackbefehle würde man vor dem Rücksprung aus einer Funktion nutzen, um den Stack
korrekt abzubauen, wenn beim Eintritt in diese Funktion folgender Stackbefehl benutzt wurde.
Tragen Sie die Lösung in die Tabelle ein:
(2 Punkte)

nach dem Eintritt in die Funktion	vor dem Austritt aus der Funktion
stmdb sp!, {r0, r1, r2, r4, lr}	
stmfd sp!, {r0-r4, r9, lr}	

Aufgabe : (8 Punkte)
Das Current Program Status Register (CPSR) hat folgende Struktur:



a) Setzen des Zero Flags auf den Wert 1 (ohne die anderen N,C,V Flags zu verändern). Das Flag Feld (Bits 24-31 vom CPSR) kann mit CPSR_f angesprochen werden.

b) Prozessor umschalten vom IRQ Mode (Bitmuster 10010) in den USER Mode (Bitmuster 10000), ohne die I,F,T Bits zu verändern. Das Kontroll Feld (Bits 0-7 vom CPSR) kann mit CPSR_c angesprochen werden.

Aufgabe : (11 Punkte)
Beantworten Sie folgende Fragen zur Prozessorarchitektur:

1. Was versteht man unter einer LOAD/STORE Architektur ? (2 Punkte)
2. Aus wieviel Stufen besteht die Pipeline des Prozessors und wie werden diese Stufen bezeichnet ? (2 Punkte)
3. Wieviel Prozessorzyklen benötigt ein Programm ausgeführt, das aus 5 Datenverabeitungsbefehlen (add, sub, mov) besteht ? (2 Punkte)
4. Wieviel Speicher benötigt ein Befehl ? (1 Punkt)
5. Aus wievielen Registern besteht ein Registersatz (ohne Statusregister) ? Welches Register ist der Program Counter (pc), Stack Pointer (sp) und das Link Register (lr) ? (2 Punkte)
6. Was versteht man unter bedingter Ausführung von Befehlen ? Geben Sie ein Beispiel mit Beschreibung an ? (2 Punkte)

Ergänzen Sie folgendes Programm um die Prozedur KopiereTabelle damit am Ende des Programms die Inhalte der Tabelle1 nach Tabelle2 kopiert sind. Denken Sie an eine ausführliche Beschreibung Ihrer Lösung (Kommentare werden auch bewertet).

```
.file "tabelle.S"

.text

.align 2

.global main

.type main,function

main:

stmfd sp!, {lr}

adr r0, [TAB1] @ Zeiger TAB1 in r0

ldr r1, [TAB2] @ Zeiger TAB2 in r1

bl KopiereTabelle

ldmfd sp!, {pc}

KopiereTabelle:                         // Den hier fehlenden Programmcode schreiben









mov pc, lr                              @ Rücksprung

TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5

TAB1ende:

TAB2: .word Tabelle2

.comm Tabelle2, TAB1ende-TAB1
```

Erg‰nzen Sie folgendes Programm um die Prozedur KopiereTabelleì damit am Ende des Programms die Inhalte der Tabelle1 nach Tabelle2 kopiert sind. Denken Sie an eine ausführliche Beschreibung Ihrer Lösung (Kommentare werden auch bewertet).

```
.file "tabelle.S"

.text

.align 2

.global main

.type main,function

main:

stmfd sp!, {lr}

adr r0, [TAB1] @ Zeiger TAB1 in r0

ldr r1, [TAB2] @ Zeiger TAB2 in r1

bl KopiereTabelle

ldmfd sp!, {pc}

KopiereTabelle:                          // Den hier fehlenden Programmcode schreiben

ldr r2, [r0] @ Länge der Tabelle in r2









mov pc, lr                               @ Rücksprung

TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5

TAB1ende:

TAB2: .word Tabelle2

.comm Tabelle2, TAB1ende-TAB1
```

Erg‰nzen Sie folgendes Programm um die Prozedur KopiereTabelleì damit am Ende des Programms die Inhalte der Tabelle1 nach Tabelle2 kopiert sind. Denken Sie an eine ausführliche Beschreibung Ihrer Lösung (Kommentare werden auch bewertet).

```
.file "tabelle.S"

.text

.align 2

.global main

.type main,function

main:

stmfd sp!, {lr}

adr r0, [TAB1] @ Zeiger TAB1 in r0

ldr r1, [TAB2] @ Zeiger TAB2 in r1

bl KopiereTabelle

ldmfd sp!, {pc}

KopiereTabelle:                         // Den hier fehlenden Programmcode schreiben

ldr r2, [r0] @ Länge der Tabelle in r2

schleife:





mov pc, lr                              @ Rücksprung

TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5

TAB1ende:

TAB2: .word Tabelle2

.comm Tabelle2, TAB1ende-TAB1
```

Erg‰nzen Sie folgendes Programm um die Prozedur KopiereTabelleì damit am Ende des Programms die Inhalte der Tabelle1 nach Tabelle2 kopiert sind. Denken Sie an eine ausführliche Beschreibung Ihrer Lösung (Kommentare werden auch bewertet).

```
.file "tabelle.S"

.text

.align 2

.global main

.type main,function

main:

stmfd sp!, {lr}

adr r0, [TAB1] @ Zeiger TAB1 in r0

ldr r1, [TAB2] @ Zeiger TAB2 in r1

bl KopiereTabelle

ldmfd sp!, {pc}

KopiereTabelle:                    // Den hier fehlenden Programmcode schreiben

ldr r2, [r0] @ Länge der Tabelle in r2

schleife:

ldr r3, [r0], #4                   @ Element aus Tabelle 1 in r3




mov pc, lr                         @ Rücksprung

TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5

TAB1ende:

TAB2: .word Tabelle2

.comm Tabelle2, TAB1ende-TAB1
```

Erg‰nzen Sie folgendes Programm um die Prozedur KopiereTabelleì damit am Ende des Programms die Inhalte der Tabelle1 nach Tabelle2 kopiert sind. Denken Sie an eine ausführliche Beschreibung Ihrer Lösung (Kommentare werden auch bewertet).

```
.file "tabelle.S"

.text

.align 2

.global main

.type main,function

main:

stmfd sp!, {lr}

adr r0, [TAB1] @ Zeiger TAB1 in r0

ldr r1, [TAB2] @ Zeiger TAB2 in r1

bl KopiereTabelle

ldmfd sp!, {pc}

KopiereTabelle:                         // Den hier fehlenden Programmcode schreiben

ldr r2, [r0] @ Länge der Tabelle in r2

schleife:

ldr r3, [r0], #4                        @ Element aus Tabelle 1 in r3

str r3, [r1], #4                        @ Element in r3 nach Tabelle 2



mov pc, lr                              @ Rücksprung

TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5

TAB1ende:

TAB2: .word Tabelle2

.comm Tabelle2, TAB1ende-TAB1
```

Erg‰nzen Sie folgendes Programm um die Prozedur KopiereTabelleì damit am Ende des Programms die Inhalte der Tabelle1 nach Tabelle2 kopiert sind. Denken Sie an eine ausführliche Beschreibung Ihrer Lösung (Kommentare werden auch bewertet).

```
.file "tabelle.S"

.text

.align 2

.global main

.type main,function

main:

stmfd sp!, {lr}

adr r0, [TAB1] @ Zeiger TAB1 in r0

ldr r1, [TAB2] @ Zeiger TAB2 in r1

bl KopiereTabelle

ldmfd sp!, {pc}

KopiereTabelle:                         // Den hier fehlenden Programmcode schreiben

ldr r2, [r0] @ Länge der Tabelle in r2

schleife:

ldr r3, [r0], #4                        @ Element aus Tabelle 1 in r3

str r3, [r1], #4                        @ Element in r3 nach Tabelle 2

subs r2, #1                             @ Zähler weniger 1


mov pc, lr                              @ Rücksprung

TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5

TAB1ende:

TAB2: .word Tabelle2

.comm Tabelle2, TAB1ende-TAB1
```

Erg‰nzen Sie folgendes Programm um die Prozedur KopiereTabelleì damit am Ende des Programms die Inhalte der Tabelle1 nach Tabelle2 kopiert sind. Denken Sie an eine ausführliche Beschreibung Ihrer Lösung (Kommentare werden auch bewertet).

```
.file "tabelle.S"

.text

.align 2

.global main

.type main,function

main:

stmfd sp!, {lr}

adr r0, [TAB1] @ Zeiger TAB1 in r0

ldr r1, [TAB2] @ Zeiger TAB2 in r1

bl KopiereTabelle

ldmfd sp!, {pc}

KopiereTabelle:                         // Den hier fehlenden Programmcode schreiben

ldr r2, [r0] @ Länge der Tabelle in r2

schleife:

ldr r3, [r0], #4                        @ Element aus Tabelle 1 in r3

str r3, [r1], #4                        @ Element in r3 nach Tabelle 2

subs r2, #1                             @ Zähler weniger 1

bne schleife                            @ weiter solange nicht 0

mov pc, lr                              @ Rücksprung

TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5

TAB1ende:

TAB2: .word Tabelle2

.comm Tabelle2, TAB1ende-TAB1
```

Schauen Sie sich das folgende Programm an und füllen Sie die Lücken in den Kommentaren beim ersten Durchlauf..

Welcher Wert wird am Ende in r0 stehen?

a - 0xffffffff

b - 0x0

c - 0x2

d - Inhalt wird nicht verändert

e - Wert von r4

f - Ist egal braucht keiner

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x
ldr r3, [r1]                    @ in r3 steht dann 0x
                                @ in r1 steht dann 0x
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x
                                @ in r1 steht dann 0x
ldr r6, [r2], #4                @ in r2 steht dann 0x
cmp r5, r6                      @ in r5 steht dann 0x
                                @ in r6 steht dann 0x
addne r0, r0, #1                @ in r0 steht dann 0x
subs r3, r3, #1                 @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                    @ in r0 steht dann 0x0
adr r1, TAB1                      @ in r1 steht dann 0x8000
adr r2, TAB2                      @ in r2 steht dann 0x
ldr r3, [r1]                      @ in r3 steht dann 0x
                                  @ in r1 steht dann 0x
label1:
ldr r5, [r1], #4                  @ in r5 steht dann 0x
                                  @ in r1 steht dann 0x
ldr r6, [r2], #4                  @ in r2 steht dann 0x
cmp r5, r6                        @ in r5 steht dann 0x
                                  @ in r6 steht dann 0x
addne r0, r0, #1                  @ in r0 steht dann 0x
subs r3, r3, #1                   @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x0
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x
ldr r3, [r1]                    @ in r3 steht dann 0x
                                @ in r1 steht dann 0x
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x
                                @ in r1 steht dann 0x
ldr r6, [r2], #4                @ in r2 steht dann 0x
cmp r5, r6                      @ in r5 steht dann 0x
                                @ in r6 steht dann 0x
addne r0, r0, #1                @ in r0 steht dann 0x
subs r3, r3, #1                 @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x0
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x8018
ldr r3, [r1]                    @ in r3 steht dann 0x
                                @ in r1 steht dann 0x
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x
                                @ in r1 steht dann 0x
ldr r6, [r2], #4                @ in r2 steht dann 0x
cmp r5, r6                      @ in r5 steht dann 0x
                                @ in r6 steht dann 0x
addne r0, r0, #1                @ in r0 steht dann 0x
subs r3, r3, #1                 @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x0
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x8018
ldr r3, [r1]                    @ in r3 steht dann 0x6
                                @ in r1 steht dann 0x8000
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x
                                @ in r1 steht dann 0x
ldr r6, [r2], #4                @ in r2 steht dann 0x
cmp r5, r6                      @ in r5 steht dann 0x
                                @ in r6 steht dann 0x
addne r0, r0, #1                @ in r0 steht dann 0x
subs r3, r3, #1                 @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x0
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x8018
ldr r3, [r1]                    @ in r3 steht dann 0x6
                                @ in r1 steht dann 0x8000
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x6
                                @ in r1 steht dann 0x8004
ldr r6, [r2], #4                @ in r2 steht dann 0x
cmp r5, r6                      @ in r5 steht dann 0x
                                @ in r6 steht dann 0x
addne r0, r0, #1                @ in r0 steht dann 0x
subs r3, r3, #1                 @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x0
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x8018
ldr r3, [r1]                    @ in r3 steht dann 0x6
                                @ in r1 steht dann 0x8000
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x6
                                @ in r1 steht dann 0x8004
ldr r6, [r2], #4                @ in r2 steht dann 0x801c
cmp r5, r6                      @ in r5 steht dann 0x
                                @ in r6 steht dann 0x
addne r0, r0, #1                @ in r0 steht dann 0x
subs r3, r3, #1                 @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                @ in r0 steht dann 0x0
adr r1, TAB1                  @ in r1 steht dann 0x8000
adr r2, TAB2                  @ in r2 steht dann 0x8018
ldr r3, [r1]                  @ in r3 steht dann 0x6
                              @ in r1 steht dann 0x8000
label1:
ldr r5, [r1], #4              @ in r5 steht dann 0x6
                              @ in r1 steht dann 0x8004
ldr r6, [r2], #4              @ in r2 steht dann 0x801c
cmp r5, r6                    @ in r5 steht dann 0x6
                              @ in r6 steht dann 0x4
addne r0, r0, #1              @ in r0 steht dann 0x
subs r3, r3, #1               @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x0
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x8018
ldr r3, [r1]                    @ in r3 steht dann 0x6
                                @ in r1 steht dann 0x8000
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x6
                                @ in r1 steht dann 0x8004
ldr r6, [r2], #4                @ in r2 steht dann 0x801c
cmp r5, r6                      @ in r5 steht dann 0x6
                                @ in r6 steht dann 0x4
addne r0, r0, #1                @ in r0 steht dann 0x1
subs r3, r3, #1                 @ in r3 steht dann 0x
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

```
.file "Vergleiche_Tabelle.S"
.text
.align 2
.global main
.type main,function
main:
stmfd sp!, {r5, r6, lr}
eor r0, r0 ,r0                  @ in r0 steht dann 0x0
adr r1, TAB1                    @ in r1 steht dann 0x8000
adr r2, TAB2                    @ in r2 steht dann 0x8018
ldr r3, [r1]                    @ in r3 steht dann 0x6
                                @ in r1 steht dann 0x8000
label1:
ldr r5, [r1], #4                @ in r5 steht dann 0x6
                                @ in r1 steht dann 0x8004
ldr r6, [r2], #4                @ in r2 steht dann 0x801c
cmp r5, r6                      @ in r5 steht dann 0x6
                                @ in r6 steht dann 0x4
addne r0, r0, #1                @ in r0 steht dann 0x1
subs r3, r3, #1                 @ in r3 steht dann 0x5
bne label1
ldmfd sp!, {r5, r6, pc}
TAB1: .word ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5
TAB1ende:
TAB2: .word 4, 1, 2, 2, 4, 5
```

Unterprogramm:

1 mov r0, #0 // Initialisierung r0

2 mov r1, #1 // Initialisierung r1

3 mov r2, #2 // Initialisierung r2

4 mov r3, #3 // Initialisierung r3

5 stmdb sp!, {r0-r3, lr}

6 stmia sp!, {r1-r2}

7 ldr pc, [sp, #12]!

Adresse	Stackspeicher	
0x820		
0x81C		← SP = 0x81C
0x818	LR	
0x814	R3	
0x810	R2	
0x80C	R1	
0x808	R0	

Aktualisieren Sie den Stack aufgrund des Blocktransferbefehls in Zeile 5, indem Sie in den Stackspeicher die entsprechenden Register eintragen.

Welchen Wert hat der Stackpointer nach

Befehl 5: 0x Befehl 6: 0x

Welchen Wert haben folgende Register nach Ausführung von Befehl 6:

R0: 0x R1: 0x

Ergänzen Sie Befehl 7 in der Form, dass das Programm an die entsprechende Stelle der rufenden Funktion zurückkehrt.

Unterprogramm:

```
1 mov r0, #0 // Initialisierung r0
2 mov r1, #1 // Initialisierung r1
3 mov r2, #2 // Initialisierung r2
4 mov r3, #3 // Initialisierung r3
5 stmdb sp!, {r0-r3, lr}
6 stmia sp!, {r1-r2}
7 ldr pc, [sp, #12]!
```

Adresse	Stackspeicher	
0x820		
0x81C		← SP = 0x81C
0x818	LR	
0x814	R3	
0x810	R2	
0x80C	R1	
0x808	R0	

Aktualisieren Sie den Stack aufgrund des Blocktransferbefehls in Zeile 5, indem Sie in den Stackspeicher die entsprechenden Register eintragen.

Welchen Wert hat der Stackpointer nach

Befehl 5: 0x808 Befehl 6: 0x

Welchen Wert haben folgende Register nach Ausführung von Befehl 6:

R0: 0x R1: 0x

Ergänzen Sie Befehl 7 in der Form, dass das Programm an die entsprechende Stelle der rufenden Funktion zurückkehrt.

Unterprogramm:

```
1 mov r0, #0 // Initialisierung r0
2 mov r1, #1 // Initialisierung r1
3 mov r2, #2 // Initialisierung r2
4 mov r3, #3 // Initialisierung r3
5 stmdb sp!, {r0-r3, lr}
6 stmia sp!, {r1-r2}
7 ldr pc, [sp, #12]!
```

Adresse	Stackspeicher	
0x820		
0x81C		SP = 0x81C
0x818	LR	
0x814	R3	
0x810	R2	
0x80C	R1	
0x808	R0	

Aktualisieren Sie den Stack aufgrund des Blocktransferbefehls in Zeile 5, indem Sie in den Stackspeicher die entsprechenden Register eintragen.

Welchen Wert hat der Stackpointer nach

Befehl 5: 0x808   Befehl 6: 0x810

Welchen Wert haben folgende Register nach Ausführung von Befehl 6:

R0: 0x   R1: 0x

Ergänzen Sie Befehl 7 in der Form, dass das Programm an die entsprechende Stelle der

rufenden Funktion zurückkehrt.

Unterprogramm:

```
1 mov r0, #0 // Initialisierung r0
2 mov r1, #1 // Initialisierung r1
3 mov r2, #2 // Initialisierung r2
4 mov r3, #3 // Initialisierung r3
5 stmdb sp!, {r0-r3, lr}
6 stmia sp!, {r1-r2}
7 ldr pc, [sp, #12]!
```

Adresse	Stackspeicher	
0x820		
0x81C		← SP = 0x81C
0x818	LR	
0x814	R3	
0x810	R2	
0x80C	R1	
0x808	R0	

Aktualisieren Sie den Stack aufgrund des Blocktransferbefehls in Zeile 5, indem Sie in den Stackspeicher die entsprechenden Register eintragen.

Welchen Wert hat der Stackpointer nach

Befehl 5: 0x808 Befehl 6: 0x810

Welchen Wert haben folgende Register nach Ausführung von Befehl 6:

R0: 0x0 R1: 0x

Ergänzen Sie Befehl 7 in der Form, dass das Programm an die entsprechende Stelle der rufenden Funktion zurückkehrt.

Unterprogramm:

```
1 mov r0, #0 // Initialisierung r0
2 mov r1, #1 // Initialisierung r1
3 mov r2, #2 // Initialisierung r2
4 mov r3, #3 // Initialisierung r3
5 stmdb sp!, {r0-r3, lr}
6 stmia sp!, {r1-r2}
7 ldr pc, [sp, #12]!
```

Adresse	Stackspeicher	
0x820		
0x81C		SP = 0x81C
0x818	LR	
0x814	R3	
0x810	R2	
0x80C	R1	
0x808	R0	

Aktualisieren Sie den Stack aufgrund des Blocktransferbefehls in Zeile 5, indem Sie in den Stackspeicher die entsprechenden Register eintragen.

Welchen Wert hat der Stackpointer nach

Befehl 5: 0x808 Befehl 6: 0x810

Welchen Wert haben folgende Register nach Ausführung von Befehl 6:

R0: 0x0 R1: 0x1

Ergänzen Sie Befehl 7 in der Form, dass das Programm an die entsprechende Stelle der rufenden Funktion zurückkehrt.

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile. Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile. Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

arm-elf-gcc Programm: Aufruf des Cross-Compilers

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile.
Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

arm-elf-gcc Programm: Aufruf des Cross-Compilers

-c Flag: nur Compilieren

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile. Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

arm-elf-gcc Programm: Aufruf des Cross-Compilers

-c Flag: nur Compilieren

-g Flag: Erzeugen von Debug Informationen

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile. Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

arm-elf-gcc Programm: Aufruf des Cross-Compilers

-c Flag: nur Compilieren

-g Flag: Erzeugen von Debug Informationen

-O2 Flag: Optimierungstufe 2

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile. Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

arm-elf-gcc Programm: Aufruf des Cross-Compilers

-c Flag: nur Compilieren

-g Flag: Erzeugen von Debug Informationen

-O2 Flag: Optimierungstufe 2

hello.c Datei: Quelle

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile.
Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

arm-elf-gcc Programm: Aufruf des Cross-Compilers

-c Flag: nur Compilieren

-g Flag: Erzeugen von Debug Informationen

-O2 Flag: Optimierungstufe 2

hello.c Datei: Quelle

Wie wird diese Make-Datei von der Konsole aufgerufen ?

Eine Make-Datei mit dem Namen MakeHello hat folgende Zeile. Erklären Sie jeden

Bestandteil dieser Zeile.

arm-elf-gcc -c -g -O2 hello.c

arm-elf-gcc Programm: Aufruf des Cross-Compilers

-c Flag: nur Compilieren

-g Flag: Erzeugen von Debug Informationen

-O2 Flag: Optimierungstufe 2

hello.c Datei: Quelle

Wie wird diese Make-Datei von der Konsole aufgerufen ?

make -f MakeHello

Das Register R1 enthalte den Binärwert

A31A30A29A28A27A26A25A24A23A22A21A20A19A18A17A16A15A14A13A12A11A10A9A8A7A6A5A4A3A2A1A0

Das Register R2 enthalte den Binärwert

B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11B10B9B8B7B6B5B4B3B2B1B0

Schreiben Sie ein ARM Assembler Programm, dass folgenden Binärwert in Register R0 schreibt:

A10A9A8A7A6A5A4A3A2A1A0B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11

Hinweis: Benutzen Sie logische Instruktionen und Shift Instruktionen.

Das Register R1 enthalte den Binärwert

A31A30A29A28A27A26A25A24A23A22A21A20A19A18A17A16A15A14A13A12A11A10A9A8A7A6A5A4A3A2A1A0

Das Register R2 enthalte den Binärwert

B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11B10B9B8B7B6B5B4B3B2B1B0

Schreiben Sie ein ARM Assembler Programm, dass folgenden Binärwert in Register R0 schreibt:

A10A9A8A7A6A5A4A3A2A1A0B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11

Hinweis: Benutzen Sie logische Instruktionen und Shift Instruktionen.

MOV R0,R2,LSR#11

Das Register R1 enthalte den Binärwert

A31A30A29A28A27A26A25A24A23A22A21A20A19A18A17A16A15A14A13A12A11A10A9A8A7A6A5A4A3A2A1A0

Das Register R2 enthalte den Binärwert

B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11B10B9B8B7B6B5B4B3B2B1B0

Schreiben Sie ein ARM Assembler Programm, dass folgenden Binärwert in Register R0 schreibt:

A10A9A8A7A6A5A4A3A2A1A0B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11

Hinweis: Benutzen Sie logische Instruktionen und Shift Instruktionen.

MOV R0,R2,LSR#11

ORR R0,R0,R1,LSL#21

Das Register R1 enthalte den Binärwert

A31A30A29A28A27A26A25A24A23A22A21A20A19A18A17A16A15A14A13A12A11A10A9A8A7A6A5A4A3A2A1A0

Das Register R2 enthalte den Binärwert

B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11B10B9B8B7B6B5B4B3B2B1B0

Schreiben Sie ein ARM Assembler Programm, dass folgenden Binärwert in Register R0 schreibt:

A10A9A8A7A6A5A4A3A2A1A0B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11

Hinweis: Benutzen Sie logische Instruktionen und Shift Instruktionen.

MOV R0,R2,LSR#11

ORR R0,R0,R1,LSL#21

oder

MOV R0,R1,LSL#21

Das Register R1 enthalte den Binärwert

A31A30A29A28A27A26A25A24A23A22A21A20A19A18A17A16A15A14A13A12A11A10A9A8A7A6A5A4A3A2A1A0

Das Register R2 enthalte den Binärwert

B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11B10B9B8B7B6B5B4B3B2B1B0

Schreiben Sie ein ARM Assembler Programm, dass folgenden Binärwert in Register R0 schreibt:

A10A9A8A7A6A5A4A3A2A1A0B31B30B29B28B27B26B25B24B23B22B21B20B19B18B17B16B15B14B13B12B11

Hinweis: Benutzen Sie logische Instruktionen und Shift Instruktionen.

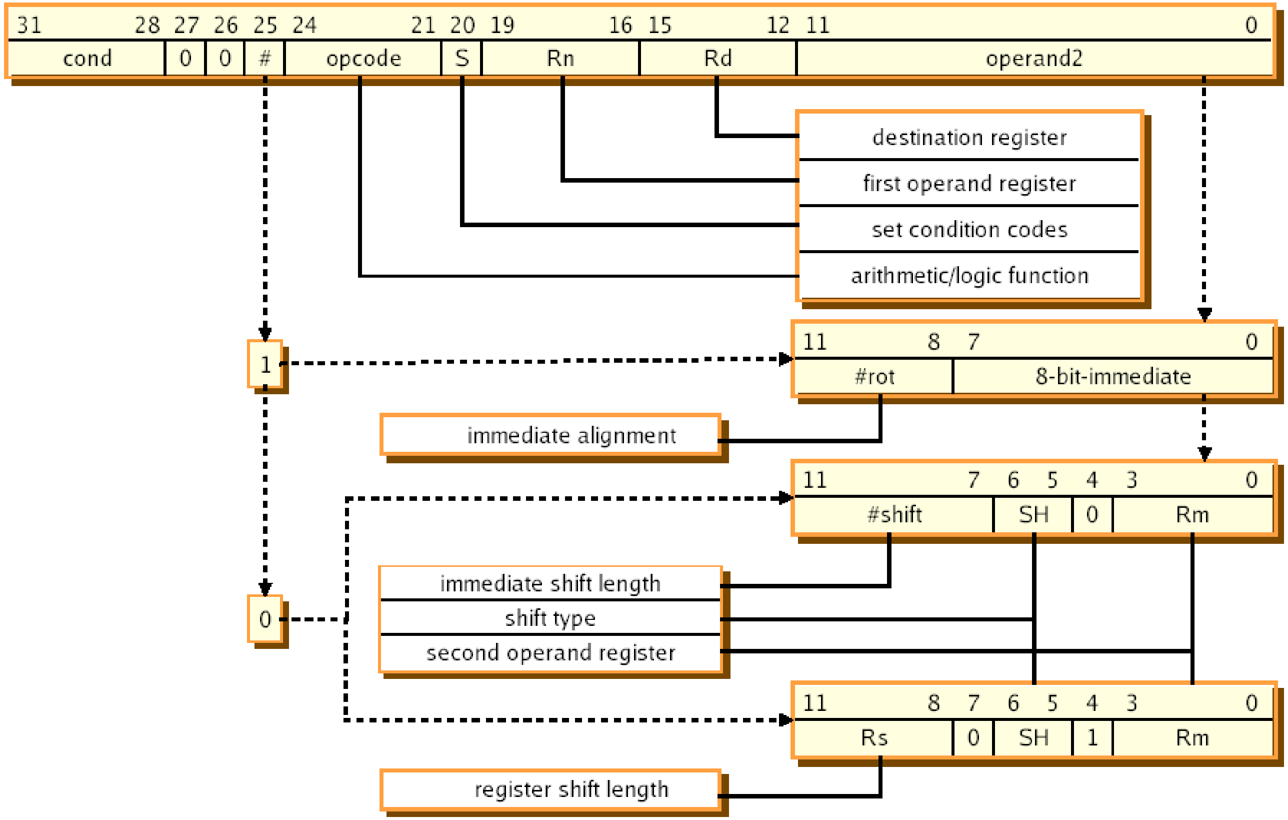
MOV R0,R2,LSR#11

ORR R0,R0,R1,LSL#21

oder

MOV R0,R1,LSL#21

ORR R0,R0,R2,LSR#11



Wie sehen folgende Befehle im Programmspeicher aus:

1. SUB R0, R1, R1

2. ADD R1, R2, R5, LSL #5

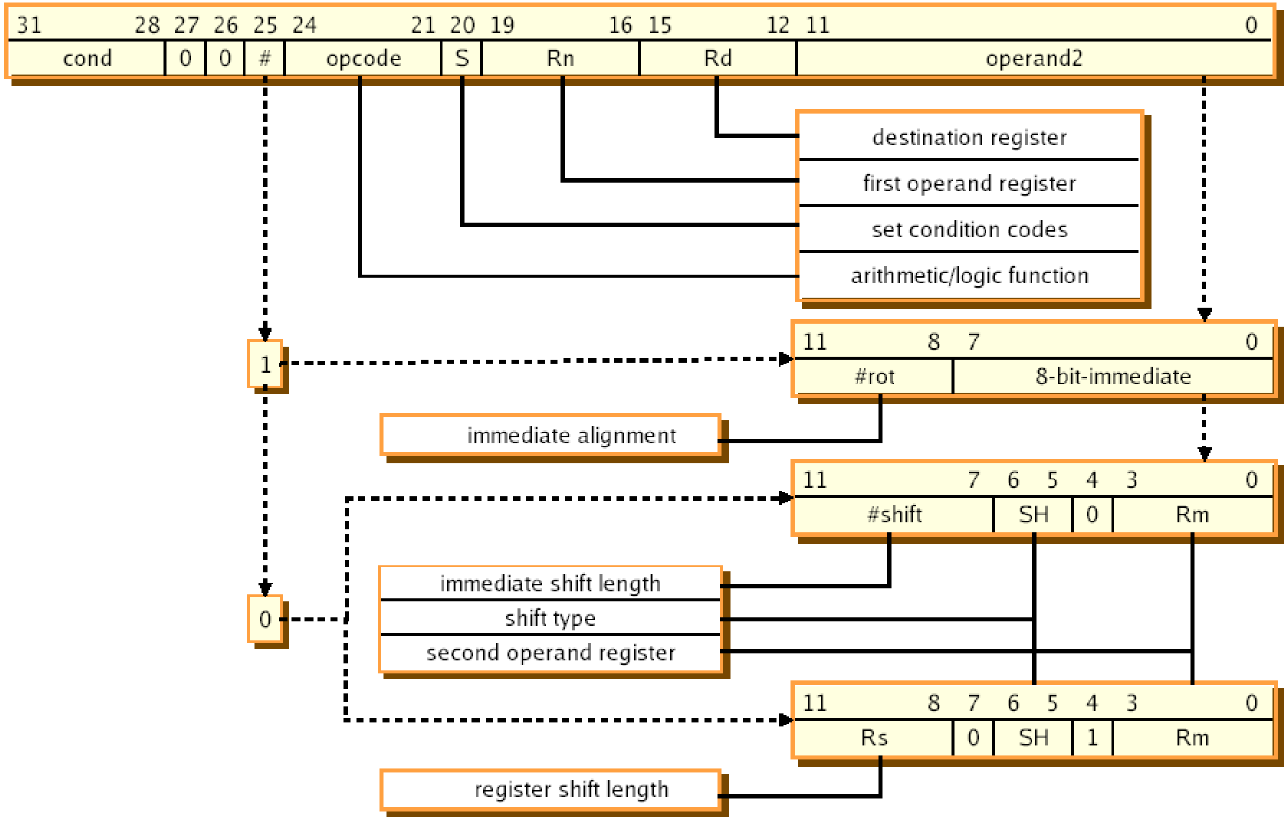
Hinweis: Opcode von SUB: 0010 Opcode von ADD: 0100

Inhalt des Condition Feldes (cond) ist für beide Befehle: 0000

Inhalt des Shift-Type Feldes (SH) ist für LSL: 00

__SUB r0, r1, r1 = 0x

__ADD r1, r2, r5, lsl #5 = 0x



Wie sehen folgende Befehle im Programmspeicher aus:

1. SUB R0, R1, R1

2. ADD R1, R2, R5, LSL #5

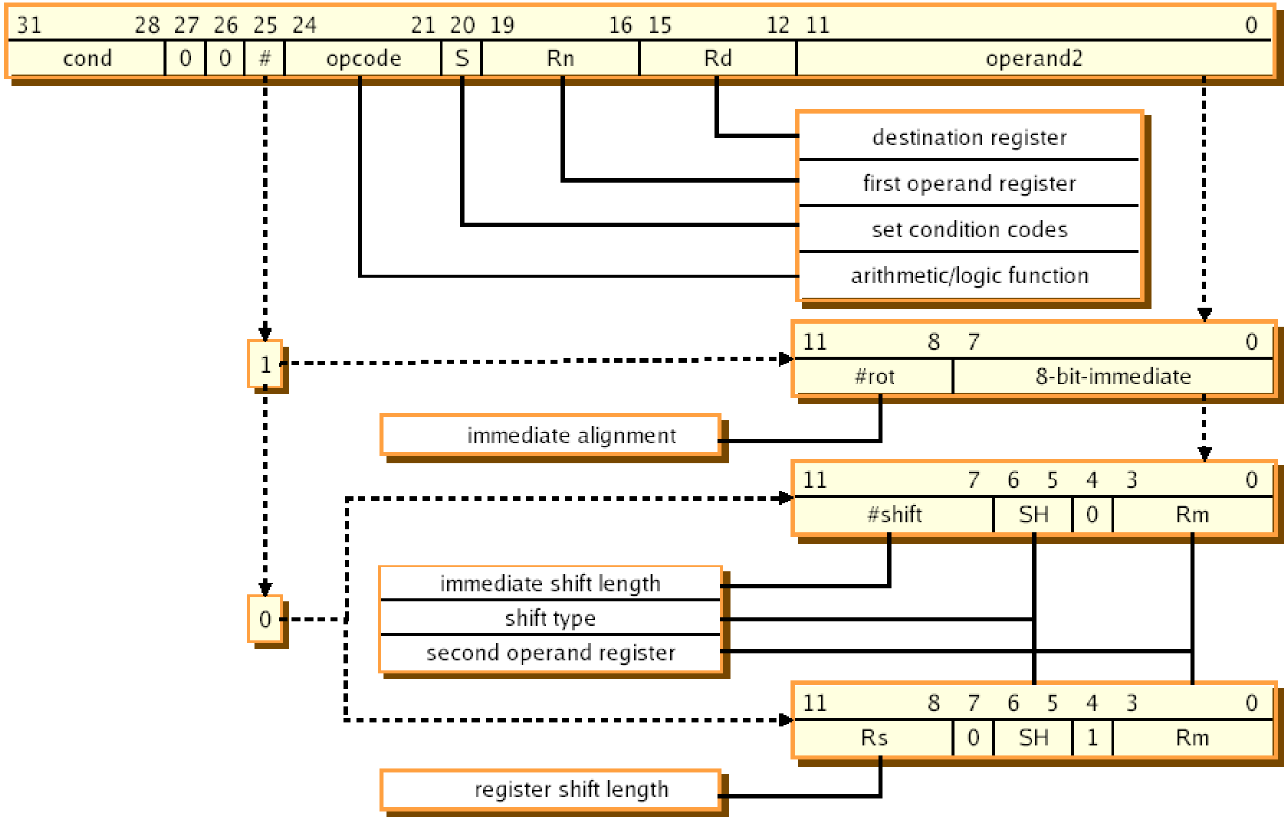
Hinweis: Opcode von SUB: 0010 Opcode von ADD: 0100

Inhalt des Condition Feldes (cond) ist für beide Befehle: 0000

Inhalt des Shift-Type Feldes (SH) ist für LSL: 00

__SUB r0, r1, r1 = 0x00410001

__ADD r1, r2, r5, lsl #5 = 0x



Wie sehen folgende Befehle im Programmspeicher aus:

1. SUB R0, R1, R1

2. ADD R1, R2, R5, LSL #5

Hinweis: Opcode von SUB: 0010 Opcode von ADD: 0100

Inhalt des Condition Feldes (cond) ist für beide Befehle: 0000

Inhalt des Shift-Type Feldes (SH) ist für LSL: 00

__SUB r0, r1, r1 = x00410001

__ADD r1, r2, r5, lsl #5 = 0x00821285

In folgender C-Routine wird die Fakultät einer Zahl rekursiv berechnet:

```
int fakultaet(int n)
{
if (n==0)
return(1);
else
return (n*fakultaet(n-1));
}
```

In folgendem Assemblerlisting sind 5 Fehler zu identifizieren:

```
fakultaet:
1 TEQ r0, #0                @ __________________
2 MOVEQ r0, #1              @ __________________
3 MOVEQ pc, lr              @ __________________
4 STMDB sp!, {r4, lr}       @ __________________
5 LDR r4, r0                @ __________________
6 SUB r0, r0, #4            @ __________________
7 B fakultaet               @ __________________
8 MLA r0, r4, r0            @ __________________
9 LDMDB sp!, {r4, pc}       @ __________________
```

Welcher Befehl wurde im berichtigten Programm vor dem Befehl der Zeile 8 ausgeführt ?

Antwort: Zeile ___

In folgender C-Routine wird die Fakultät einer Zahl rekursiv berechnet:

```
int fakultaet(int n)
{
if (n==0)
return(1);
else
return (n*fakultaet(n-1));
}
```

In folgendem Assemblerlisting sind 5 Fehler zu identifizieren:

```
fakultaet:
1 TEQ r0, #0                @ __________________
2 MOVEQ r0, #1              @ __________________
3 MOVEQ pc, lr              @ __________________
4 STMDB sp!, {r4, lr}       @ __________________
5 LDR r4, r0                @ MOV r4, r0______
6 SUB r0, r0, #4            @ __________________
7 B fakultaet               @ __________________
8 MLA r0, r4, r0            @ __________________
9 LDMDB sp!, {r4, pc}       @ __________________
```

Welcher Befehl wurde im berichtigten Programm vor dem Befehl der Zeile 8 ausgeführt ?

Antwort: Zeile ___

In folgender C-Routine wird die Fakultät einer Zahl rekursiv berechnet:

```
int fakultaet(int n)
{
if (n==0)
return(1);
else
return (n*fakultaet(n-1));
}
```

In folgendem Assemblerlisting sind 5 Fehler zu identifizieren:

```
fakultaet:
1 TEQ r0, #0                @ __________________
2 MOVEQ r0, #1              @ __________________
3 MOVEQ pc, lr              @ __________________
4 STMDB sp!, {r4, lr}       @ __________________
5 LDR r4, r0                @ MOV r4, r0______
6 SUB r0, r0, #4            @ SUB r0, r0, #1
7 B fakultaet               @ __________________
8 MLA r0, r4, r0            @ __________________
9 LDMDB sp!, {r4, pc}       @ __________________
```

Welcher Befehl wurde im berichtigten Programm vor dem Befehl der Zeile 8 ausgeführt ?

Antwort: Zeile ___

In folgender C-Routine wird die Fakultät einer Zahl rekursiv berechnet:

```
int fakultaet(int n)
{
if (n==0)
return(1);
else
return (n*fakultaet(n-1));
}
```

In folgendem Assemblerlisting sind 5 Fehler zu identifizieren:

```
fakultaet:
1 TEQ r0, #0              @ __________________
2 MOVEQ r0, #1            @ __________________
3 MOVEQ pc, lr            @ __________________
4 STMDB sp!, {r4, lr}     @ __________________
5 LDR r4, r0              @ MOV r4, r0_____
6 SUB r0, r0, #4          @ SUB r0, r0, #1
7 B fakultaet             @ BL fakultaet___
8 MLA r0, r4, r0          @ __________________
9 LDMDB sp!, {r4, pc}     @ __________________
```

Welcher Befehl wurde im berichtigten Programm vor dem Befehl der Zeile 8 ausgeführt ?

Antwort: Zeile ___

In folgender C-Routine wird die Fakultät einer Zahl rekursiv berechnet:

```
int fakultaet(int n)
{
if (n==0)
return(1);
else
return (n*fakultaet(n-1));
}
```

In folgendem Assemblerlisting sind 5 Fehler zu identifizieren:

```
fakultaet:
1 TEQ r0, #0                 @ _________________
2 MOVEQ r0, #1               @ _________________
3 MOVEQ pc, lr               @ _________________
4 STMDB sp!, {r4, lr}        @ _________________
5 LDR r4, r0                 @ MOV r4, r0_____
6 SUB r0, r0, #4             @ SUB r0, r0, #1
7 B fakultaet                @ BL fakultaet___
8 MLA r0, r4, r0             @ MUL r0,r4,r0__
9 LDMDB sp!, {r4, pc}        @ _________________
```

Welcher Befehl wurde im berichtigten Programm vor dem Befehl der Zeile 8 ausgeführt ?

Antwort: Zeile ___

In folgender C-Routine wird die Fakultät einer Zahl rekursiv berechnet:

```
int fakultaet(int n)
{
if (n==0)
return(1);
else
return (n*fakultaet(n-1));
}
```

In folgendem Assemblerlisting sind 5 Fehler zu identifizieren:

```
fakultaet:
1 TEQ r0, #0                @ __________________
2 MOVEQ r0, #1              @ __________________
3 MOVEQ pc, lr              @ __________________
4 STMDB sp!, {r4, lr}       @ __________________
5 LDR r4, r0                @ MOV r4, r0_____
6 SUB r0, r0, #4            @ SUB r0, r0, #1
7 B fakultaet               @ BL fakultaet___
8 MLA r0, r4, r0            @ MUL r0,r4,r0__
9 LDMDB sp!, {r4, pc}       @ LDMIA sp!, {r4, pc}
```

Welcher Befehl wurde im berichtigten Programm vor dem Befehl der Zeile 8 ausgeführt ?

Antwort: Zeile ___

In folgender C-Routine wird die Fakultät einer Zahl rekursiv berechnet:

```
int fakultaet(int n)
{
if (n==0)
return(1);
else
return (n*fakultaet(n-1));
}
```

In folgendem Assemblerlisting sind 5 Fehler zu identifizieren:

```
fakultaet:
1 TEQ r0, #0                @ __________________
2 MOVEQ r0, #1              @ __________________
3 MOVEQ pc, lr              @ __________________
4 STMDB sp!, {r4, lr}       @ __________________
5 LDR r4, r0                @ MOV r4, r0_____
6 SUB r0, r0, #4            @ SUB r0, r0, #1
7 B fakultaet               @ BL fakultaet___
8 MLA r0, r4, r0            @ MUL r0,r4,r0__
9 LDMDB sp!, {r4, pc}       @ LDMIA sp!, {r4, pc}
```

Welcher Befehl wurde im berichtigten Programm vor dem Befehl der Zeile 8 ausgeführt ?

Antwort: Zeile _3_