Mikroprozessorsysteme Praktikum 1

Aufgabe 1

```
int main (void)
{
        int a = 0x1;
        int b = 0x2;

        int *reg = &a;
        int *reg2 = &b;
        return (0);
}
```

Wo liegen die Werte der Variabel im Speicher?
Sie liegen im Stack
a = 0x203fff0 (int *reg = &a)
b = 0x203ffec (int *reg2 = &b)

Wie kommen die Werte in den Speicher?
PUSH {r11}
ADD r11, sp, #0;
SUB sp, sp, #16;
MOV r3, #1;
STR r3, [r11, #-12];
MOV r3, #2;
STR r3, [r11, #-16]
…

Welche Register werden verwendet?
R3 für die Integer Werte

OP = 0 → alles wird normal ausgeführt
OP = 1 → alles wird wegoptimiert
OP = 2 → alles wird wegoptimiert

Aufgabe 2

```
int a = 0x1;
int b = 0x2;
int c;

int main (void)
{
        int *reg = &a;
        int *reg2 = &b;
        return (0);
}
```

Was ändert sich?
Sie werden vorher auf dem Heap abgelagert

Warum?
Weil lokale Variabeln auf dem Stack gelagert werden und globale Variabeln auf dem Heap gelagert werden.
Wo werden die Werte der Variabel gespeichert?
Sie werden im Heap gespeichert.

Aufgabe 3

```
volatile int a = 0x1;
volatile int b = 0x2;


int main (void)
{
        int c = 0x3;
        int d = 0x4;

        int *reg = &a;
        int *reg2 = &b;

        a = c;
        d = b;

        return (0);
}
```

Was stellen Sie fest?
Ohne volatile werden Variabeln die initialisiert werden, wegoptimiert bzw. übersprungen.

OP = 0 → alles wird normal ausgeführt
OP = 1 → optimiert alles weg bis auf (lokale Variabel = globale Variabel)
OP = 2 → optimiert alles weg, jedoch ruft er (lokale Variabel = globale Variabel) auf nachdem return.

Aufgabe 4

```
volatile int a = 1;
volatile int b = 1;

int addition(int, int, int);

int main (void)
{
        int c = 3;
        int d = 4;

        addition(a,c,d);

        return (0);
}

int addition(int reg1, int reg2, int reg3)
{
        int sum = reg1 + reg2 + reg3;

        return sum;
}
```

OP = 0 → alles wird normal ausgeführt
OP = 1 → alles wird wegoptimiert, jedoch kann man einzelne Variabeln auf volatile setzen, damit dies verhindert wird.
OP = 2 → lokale Variabel werden nicht in der Rechnung miteingebunden, also sum = 1;

Aufgabe 5

```
// Loesung zu Termin1
// Aufgabe 5
// Namen: ___________; ___________
// Matr.: ___________; ___________
// vom:   ___________

// Beispiel des Anlegens und der Nutzung einer Zeigervariablen
#define PIOB_PER ((unsigned int *) 0xFFFF0000)

// Global angelegte Variable mit der Adresse fuer PIOB_CODR
unsigned int adr_PIOB_CODR = 0xFFFF0034;

int main (void)
{
// Variable mit der Adresse fuer PIOB_OER
  unsigned int adr_PIOB_OER = 0xFFFF0010;
// PIOB_PER = 0x100
  *PIOB_PER = 0x100;
// PIOB_OER = 0x100;
  *((unsigned int *) adr_PIOB_OER) = 0x100;

  while (1)
  {
// PIOB_SODR = 0x100;
  *((unsigned int *) 0xffff0030) = 0x100;


// PIOB_CODR = 0x100;
  *((unsigned int *) adr_PIOB_CODR) = 0x100;


  }

  return (0);
}
```

Welche Variante würden Sie bevorzugen und warum?
Mit Optimierung 1: Da Speicherplatz nicht verschwndet wird. Die Werte, die nicht benutzt werden, werden rausoptimiert ???

Aufgabe 6

Wie viele Byte Speicher benötigen Sie für die Initialisierung und um die LED in einer Endlosschleife blinken zu lassen?
Optimierungsstufe 3