Aufgabe 1

Um das kleinste gemeinsame Vielfache der Zahlen m und n zu berechnen, berechnen wir die Vielfachen beider Zahlen, und wählen das erste Vielfache aus, das bei beiden Zahlen auftaucht. Damit wir sofort merken, wenn wir das Ergebnis gefunden haben, erhöhen wir immer nur die kleinere Zahl.

(a) Abbildung 1 zeigt diesen Algorithmus als Programmablaufplan:

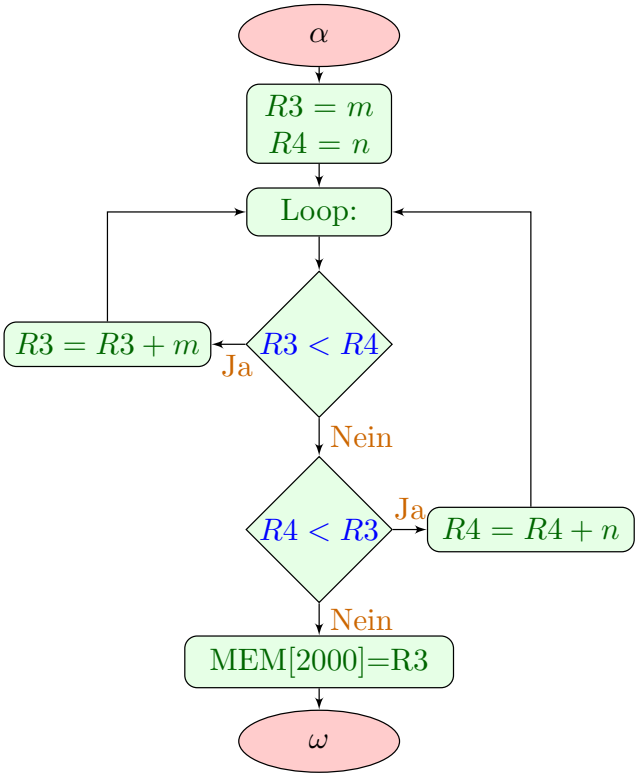

Abbildung 1: Flussdiagramm

Register	Funktion
R1,R2	m, n Die Register enthalten die beiden Parameter und werden von unserem Programm nicht verändert.
R3,R4	Vielfache von m bzw. n In den Registern werden die Vielfachen von m und n hochgezählt, bis sie die gleiche Zahl enthalten. Dann wird eines der beiden gespeichert.
R5	Hilfsregister für bedingte Sprünge

Tabelle 1: Registerbelegung

(b) Als DLX-Code lässt sich das Programm also beispielsweise folgendermaßen schreiben:

```
      ADD   R3, R1, R0     / Initialisiere das m-Vielfache mit m
      ADD   R4, R2, R0     / Initialisiere das n-Vielfache mit n
Loop: SLT   R5, R3, R4     / Wenn m-Vielfache < n-Vielfache:
      BNEZ  R5, AddM       / Erhoehe das Vielfache von m um ein weiteres m.
      SLT   R5, R4, R3     / Wenn n-Vielfache < m-Vielfache:
      BNEZ  R5, AddN       / Erhoehe das Vielfache von n um ein weiteres n.
                           / Sonst sind die Vielfache gleich und somit das
                              kgV.
      SW    2000(R0), R3   / Also speichere das Ergebnis.
      HALT

AddM: ADD   R3, R3, R1     / Erhoehe das m-Vielfache um m
      J     Loop           / Wiederhole die Schleife
AddN: ADD   R4, R4, R2     / Erhoehe das n-Vielfache um n
      J     Loop           / Wiederhole die Schleife
```

(c) Die DLX-Implementierung des Programms erhält zwei Parameter über die Register R1 und R2. Sie nutzt die Register R3 bis R5 für interne Berechnungen. Um das Programm als Unterprogramm nutzen zu können, müssen daher die Registerbelegungen von R3 bis R5 vorher auf dem Stack gespeichert werden. Zusätzlich muss ein Label in der ersten Zeile (z.B. `Kgv:`) eingefügt werden, zu dem für den Programmstart gesprungen werden kann. Statt `HALT` muss mit dem Return-Befehl `JR R31` zum Hauptprogramm zurückgesprungen werden:

```
Kgv:  SW 0(R30), R3       / Sichere Register R3, R4, R5 auf Stack
      ADDI R30, R30, #4   / und erhoehe jeweils ToS
      SW 0(R30), R4
      ADDI R30, R30, #4
      SW 0(R30), R5
      ADDI R30, R30, #4

      ADD   R3, R1, R0     / Initialisiere das m-Vielfache mit m
      ADD   R4, R2, R0     / Initialisiere das n-Vielfache mit n
Loop: SLT   R5, R3, R4     / Wenn m-Vielfache < n-Vielfache:
      BNEZ  R5, AddM       / Erhoehe das Vielfache von m um ein weiteres m.
      SLT   R5, R4, R3     / Wenn n-Vielfache < m-Vielfache:
      BNEZ  R5, AddN       / Erhoehe das Vielfache von n um ein weiteres n.
                           / Sonst sind die Vielfache gleich und somit das
                              kgV.
      SW    2000(R0), R3   / Also speichere das Ergebnis.

      SUBI R30, R30, #4     / Lade Register R5, R4, R3 von Stack in
         umgekehrter Reihenfolge
```

```
      LW R5, 0(R30)        / und reduziere jeweils ToS
      SUBI R30, R30, #4
      LW R4, 0(R30)
      SUBI R30, R30, #4
      LW R3, 0(R30)
      JR R31             / Springe zurueck zu Hauptprogramm

AddM: ADD  R3, R3, R1    / Erhoehe das m-Vielfache um m
      J    Loop          / Wiederhole die Schleife
AddN: ADD  R4, R4, R2    / Erhoehe das n-Vielfache um n
      J    Loop          / Wiederhole die Schleife
```

Alternativ kann die Übergabe der Parameter auch über den Stack erfolgen. Beide Parameter liegen also oben auf dem Stack, es müssen nun die fünf Register R1 bis R5 gesichert werden:

```
Kgv:  SW 0(R30), R1      / Sichere Register R1-R5 auf Stack
      SW 4(R30), R2
      SW 8(R30), R3
      SW 12(R30), R4
      SW 16(R30), R5
      LW R1,-8(R30)      / Lade m von Stack
      LW R2,-4(R30)      / Lade n von Stack
      ADDI R30, R30, #20 / und erhoehe ToS

      ADD  R3, R1, R0    / Initialisiere das m-Vielfache mit m
      ADD  R4, R2, R0    / Initialisiere das n-Vielfache mit n
Loop: SLT  R5, R3, R4    / Wenn m-Vielfache < n-Vielfache:
      BNEZ R5, AddM      / Erhoehe das Vielfache von m um ein weiteres m.
      SLT  R5, R4, R3    / Wenn n-Vielfache < m-Vielfache:
      BNEZ R5, AddN      / Erhoehe das Vielfache von n um ein weiteres n.
                         / Sonst sind die Vielfache gleich und somit das
                            kgV.
      SW   2000(R0), R3  / Also speichere das Ergebnis.

      LW R5, -4(R30)     / Stelle Registerbelegung des Hauptprogramms
      LW R4, -8(R30)     / wieder her
      LW R3, -12(R30)
      LW R2, -16(R30)
      LW R1, -20(R30)
      SUBI R30, R30, #20 / und reduziere ToS

      JR R31             / Springe zurueck zu Hauptprogramm

AddM: ADD  R3, R3, R1    / Erhoehe das m-Vielfache um m
      J    Loop          / Wiederhole die Schleife
```

```
AddN:  ADD   R4, R4, R2     / Erhoehe das n-Vielfache um n
       J     Loop           / Wiederhole die Schleife
```

Aufgabe 2

`LW R3, 1000(R1)`: LW ist ein Immediate-Befehl. In der IF-Phase wird der Befehl ins Instruction Register geladen. Ebenso wird hier bereits der neue Program Counter für den nächsten Befehl (Aktueller PC + 4) errechnet. In der ID-Phase wird R1 in das Operandenregister A geladen. Der Immediate-Wert 1000 wird über die Sign-Extend-Einheit von 16 auf 32 Bit erweitert, um ihn als zweiten Operand für die ALU nutzen zu können. In der EX-Phase werden Operand A und der Immediate-Wert addiert und somit die Speicheradresse berechnet. In der MEM-Phase wird der an der errechneten Adresse liegende Speicherinhalt aus dem Data Memory ausgelesen. In der WB-Phase wird dieser Wert in das Register R3 geschrieben.