Aufgabe 1: Fließbandverarbeitung

Sorgen Sie dafür dass die folgende Befehlsfolge aus Pseudoinstructions in den jeweiligen Pipelines konfliktfrei ausgeführt wird:

```
add r0, r1, r2
sub r1, r5, r0
mov [rsp+8], r5
or  r0, r5, r4
mov r3, [rsp+24]
and r1, r0, r3
add r0, r1, r3
add r0, r0, 0x341D
add r0, r0, 0x52F6
```

Hinweis: `add a, b, c` *entspricht* `a=b+c`

Einfache Pipeline Gehen Sie hier von einer einfachen 5-stufigen Pipeline aus: Befehl holen (IF), Befehl dekodieren (ID), Operanden holen (OF), Ausführung (EX), Rückspeichern (WB). Weiterhin liegt eine reine Load/Store-Architektur ohne architekturelle Beschleunigungsmaßnahmen (z. B. Forwarding, Reordering etc.) oder Hardware zur Erkennung von Hemmnissen vor. Operanden können erst dann aus Registern geholt werden, nachdem sie zurück gespeichert wurden.

Verbesserte Pipeline Gehen Sie jetzt davon aus, dass die Pipeline aus dem vorherigen Aufgabenteil die in der Vorlesung kennengelernten Forwarding-Varianten/Shortcuts verwendet.

Aufgabe 2: Arithmetik

Gegeben Sei folgende Formel:

$$\frac{((a+b)\cdot(c-d))\cdot(e\cdot 8+f\cdot 4-g\div 2+h\div 4)}{3}$$

Integer-Arithmetik Implementieren Sie eine Funktion, die den ganzzahliegen Anteil der Formel berechnet. Die Funktion soll folgende Signatur haben:

```
int32_t formula_int(int32_t a, int32_t b, int32_t c, int32_t d,
                    int32_t e, int32_t f, int32_t g, int32_t h);
```

Stellten Sie sich folgende Fragen:

1. Wie kommen Sie an die Parameter `g` und `h` ran?
2. Was müssen Sie vor den Divisionen beachten?
3. Was passiert wenn Sie sehr große Werte ($\gg 2.000.000.000$) als Parameter übergeben?

Hinweis: Beachten Sie den Datentyp, signed 32-Bit Integer.

Optional: Optimieren Sie Ihre Funktion, in dem Sie Shiftbefehle (SHR, SHL, SAR, SAL) verwenden.

Fließkomma-Arithmetik Implementieren Sie die obige Formel für Fließkommazahlen. Die Funktion soll nun folgende Signatur haben:

```
double formula_flt(double a, double b, double c, double d,
                   double e, double f, double g, double h);
```

Machen Sie sich dazu mit der SSE-Unit Ihres Prozessors vertraut.

1. Welche zusätzlichen Register stellt diese bereit?
2. Welche neuen Befehle benötigen Sie?
3. Wie ist die Calling Convention für Floats?

Achtung: Sie können Floating-Point Zahlen (bspw. $3,0$) nicht als Immediate Value benutzen, sondern müssen diese vorher im Speicher ablegen und dann mit Lade-Operatoren diese auslesen. Mittlerweile wird aus sicherheitstechnischen Gründen der direkte Zugriff auf Speicherkonstanten eingeschränkt. Stattdessen wird sogenannte IP-Relative Addressierung genutzt, in NASM wird dafür bei der Adressierung der Präfix `rel` benötigt:

```
SECTION .data
meine_konstante: DQ 42.0
; ...
MULSD xmm0, [rel meine_konstante]
```

Alternativ kann man mit der Linkeroption `-no-pie` dieses Feature für das eigene Programm ausschalten.