Benke Hargitai 5370932
Lukas Seyfried 5343019

1	2	3	Σ
8	6	6	20

Finde es toll, dass ihr im Tutorat so gut mitarbeitet n_n

sehr gute Arbeit! n_n

Aufgabenblatt 02

Abgabe: 04.11.2022

8/8

Aufgabe 1

uh und schön, dass ihr das auch noch so schön formatiert, unterschiedliche Schrifttarten

```
STOREIN ACC SP i:
ACCLd, IRd, ALUAd, ASMd, SPDd
```

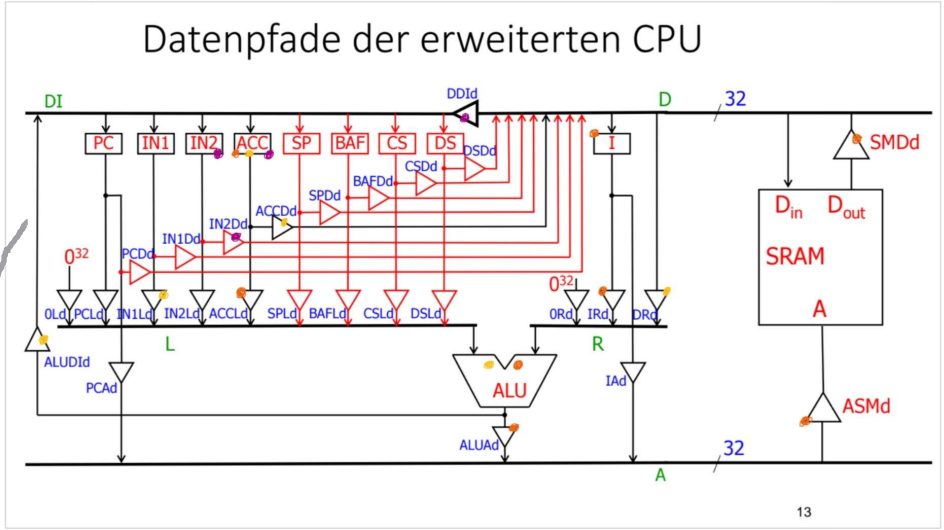
Der Befehl speichert den Inhalt des SP-Registers an die Adresse ACC+i im Hauptspeicher. Wir benötigen also die Treiber um den Inhalt von ACC und I auf die ALU anzulegen, (ACCLd, IRd), die Treiber um die so berechnete Speicheradresse an den SRAM anzulegen (ALUAd, ASMd), sowie den Treiber um den Stackpointer mit dem Datenbus zu verbinden (SPDd).

```
MOVE IN2 ACC:
IN2Dd, DDId
```

Der Befehl kopiert den Inhalt des IN2-Registers in den Akkumulator. Mit der ReTi-Erweiterung werden hierfür nurnoch 2 Treiber benötigt: IN2Dd um den zu kopierenden Wert auf den Datenbus zu liefern, sowie DDId um ihn von dort in den Akkumulator zu kopieren.

```
ADD ACC IN1:
ACCDd, DRd, IN1Ld, ALUDId
```

Der Befehl addiert die Werte aus ACC und IN1 und schreibt das Ergebnis wieder in den Akkumulator. Hierzu muss also eines der Register auf den linken Operanden-Bus und das andere über den Datenbus auf den rechten Operanden-Bus gelegt werden. Die Summer wird über den ALUDId-Treiber in den Akkumulator geschrieben.



a) STOREIN ACC SP i
b) MOVE IN2 ACC
c) ADD ACC IN1

Der Operand i im Befehl LOADI D i besteht wegen der Befehls-Kodierung nicht aus den vollen 32 Bit. Diese können mit der 0 auf dem linken Operanden-Bus gefüllt werden.

Aufgabe 2

6/6

push:

```
STOREIN SP ACC 0
SUBI SP1
```

pop:

```
ADDI SP 1
LOADIN SP ACC 0
```

Mit dem restlichen ReTi-Befehlssatz ist ein Sprung an eine feste aber beliebige Speicheradresse nicht ohne Weiteres möglich, da die existierenden JUMP-Befehle nur relative Sprünge erlauben.

man kann auch direkt das PC-Register manipulieren

Da die Aufgabe seltsam gestellt ist, gibt es hierfür trotz allem volle Punkte.

6/6

Aufgabe 3

brk(0x... Der Command `brk` dient an dierer Stelle zum Erstellen eines neuen Speicherplatzes (`int value;`).

fopen Zu `fopen` können wir die Befehle `openat` und `fstat` zuordnen.

fprintf Zu `fprintf` gehört der Befehl `write`.

fclose Zu `fclose` gehört `close`.

mit exit_group wird thread beendet

fscanf Zu `fscanf`, der Befehl `read`.

return Und zu `return` gehört schließlich `exit_group`.

nicht wirlich, aber passt schon

Begründung: Wenn wir `strace ./test` ausführen, kommen die System Calls genau in der Reihenfolge, wie die Commands in `test.c`.

Außerdem haben wir die manpages angeschaut und logisch ausgewählt, was wozu gehören könnte.

nice kurz und knapp damit auseinandergesetzt, ohne rießige Textberge, top! ^_^

eigentlich sollten Beschreibungstexte was welcher Befehl macht dazugeschrieben werden. Das ist allerdings meiner Meinung nach Zeitverschwendung, daher ziehe ich dafür keine Punkte ab, da das zurodnen ja voraussetzt, dass ihr irgendwie verstanden habt, was die Systemcalls ungefähr machen