Betriebssysteme Übungsblatt 02

Anne Ross und Diana Hörth

Oktober 2022

Aufgabe 1

a) - STOREIN ACC SP i

- IRd
- SPLd
- ALUAd
- (ACCDd)

-0.25 es muss ACCLd sein, weil ACC mit i verrechnet werden muss

Der Treiber IRd läd die Variabel i in die ALU und SPLd popt den Stack und läd es ebenfalls in die ALU. Danach wird der Inhalt in den ACC geladen.

b) - MOVE IN2 ACC

- ALUDId
- IN2Ld
- 0Rd

-0.25 es war leider nach dem kürzesten Weg gefragt. Über die ALU zu gehen und das 0Rd Register aktivieren zu müsssen ist nicht der kürzeste Weg. Der kürzese Weg ist über In2Dd und DDId.

IN2Ld lädt den Inhalt des Indexregister 2 in die ALU und 0Rd ladet eine 0 in die ALU. Nun wird der Inhalt der ALU mit ALUDId in den ACC geschoben.

c) - ADD ACC IN1

- DRd
- ACCDd
- ALUDId
- IN1Ld

Mit ACCDd wird der ACC in den rechten Datenbuss geladen und mit DRd in die AlU. Jetzt ladet IN1Ld das Indexregister 1 in die ALU. Das Ergebnis wird mit ALUDId in den ACC geladen.

d)

kleine Korrektur: LOADI Reg i; LOADI i gibt es nicht

Wenn man den Befehl "LOADI i" ausführt gibt es bei dem linken Operanten-Buss eine 0, da nur auf i zugegriffen werden soll und man keine Source benötigt.

Aufgabe 2

a)

push()

PC	Befehl	Kommentare
0	SUBI SP 1	Der SP wird um eins nach oben verschoben, sodass er auf eine weitere leere Zelle zeigt.
1	STOREIN SP ACC 1	Das was im ACC liegt wird an die Stelle auf die der SP zuvor gezeigt hat geladen.

pop()

PC	Befehl	Kommentare
0	LOADIN SP Acc 1	Inhalt der ersten befüllten Zelle im Stack ins Indexregister 1 laden.
1	ADDI SP 1	Den Stackpointer um eins nach unten in dem Stack bewegen
3	LOAD ACC IN1	Laden das Inhaltes des Indexregister 1 in ACC

wozu?

im In1 Register habt ihr doch schon stehen, was ihr braucht

Warum nicht direkt LOADIN SP ACC 1
-0.25 LOAD ACC IN1 gibt es leider nicht als Befehl, ihr meint Move IN1 Acc, oder?

b)

RTI (return from interrupts) musste eingeführt werden, da man sonst jedes mal den Befehl schreiben müsste um wieder in den usermodus zu kommen, aber da es sich am Ende der Betriebssystemroutine befindet, passiert es von allein und die Programme können ganz normal weiter laufen.
INT i (i steht für die jeweilige Interruptnummer) wird benötigt, da es verschiedene Arten von Interrupts gibt und ihnen allgemein Zahlen zuzuordnen ist einfach als sie jedes mal neu zu definieren.

es gibt verschiedene Interrupt Service Routinen

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

es gibt nur Hardware- und Software-Interrupt. Int i ist ein Software-Interrupt

Aufgabe 3

```
brk(0x559176c3b000)                     = 0x559176c3b000
openat(AT_FDCWD, "myfile.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fstat(3, {st_mode=S_IFREG|0664, st_size=0, ...}) = 0
write(3, "0 1 2 3 4 5 6 7 8 9 10 11 12 13 "..., 4096) = 4096
write(3, "041 1042 1043 1044 1045 1046 104"..., 4096) = 4096
write(3, "60 1861 1862 1863 1864 1865 1866"..., 3198) = 3198
close(3)                                = 0
openat(AT_FDCWD, "myfile.txt", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0664, st_size=11390, ...}) = 0
read(3, "0 1 2 3 4 5 6 7 8 9 10 11 12 13 "..., 4096) = 4096
read(3, "041 1042 1043 1044 1045 1046 104"..., 4096) = 4096
read(3, "60 1861 1862 1863 1864 1865 1866"..., 4096) = 3198
read(3, "", 4096)                       = 0
close(3)                                = 0
exit_group(42)                          = ?
+++ exited with 42 +++
```

Figure 1: Die Zeilennummer entspricht der Listennummer

1. brk() verzögert den Abbruch des Programms, sodass die einzelnen Schritte länger im Speicher bleiben und abgerufen werden können
2. Die Textdatei "myfile.txt" wurde geöffnet und 3 als Bezeichnung zurückgegeben. Es wird in die Datei rein geschrieben.
3. Mit fstat() werden die Informationen zu unserer Datei (3) abgefragt und gibt 0 zurück.
4. write() schreibt in die Textdatei und zwar schreibt es bytes aus dem Zwischenspeicher beginnend mit dem Zeiger der zur Textdatei verweist. Die Anzahl an geschriebener bytes muss nicht der Anzahl in Befehl entsprechen (4096), wenn der Aufruf durch ein Interrup unterbrochen wird.
5. Hier passiert das selbe wie oben bloß hat sich der Zeiger geändert. passiert das selbe wie oben bloß hat sich der Zeiger geändert und die Anzahl geändert.
6. Bei close() wird der "file descripter" geschlossen, in unserem Fall die wo in der zweiten Zeile geöffnet wurde, sodass eine neue Datei angeschaut werden kann. Gibt 0 zurück, da Programm beendet wurde.
7. Die Datei "myfile.txt" wird geöffnet und der 3 zugeordnet. Es wird nur von der Datei abgelesen und nichts verändert.
8. Die Informationen zu der Datei werden abgefragt und 0 zurück gegeben.
9. read() liest die Datei und gibt die Anzahl der Bytes zurück.

10. ” ”
11. ” ”
12. Wenn die Datei zuende ist gibt read() eine 0 zurück.
13. Der ”file descripter” wird wieder geschlossen und gibt die 0 zurück.
14. 42 wird zuruück gegeben.
15. Das ende des Programms wurde erreicht und damit wird es beendet.