

## Blatt 2

**Aufgabe 1.** In der Vorlesung haben sie eine veränderte Form der aus TI bekannten ReTI kennengelernt.

- a) Nehmen Sie an, auf der erweiterten ReTI wird der Befehl

  
`STORE ACC SP i`

ausgeführt. Betrachten Sie nur die Execute-Phase. Welche Treiber müssen hierfür aktiv geschaltet werden? Begründen Sie Ihre Antwort.

- b) Diesmal wird auf der erweiterten ReTI der Befehl

`MOVE IN2 ACC`

ausgeführt. Betrachten Sie wieder nur die Execute-Phase. Geben Sie die minimale Anzahl an Treibern an, die für den Befehl aktiviert werden müssen und benennen Sie diese Treiber. Begründen Sie Ihre Antwort.

- c) Die Befehlsklasse der COMPUTE-Befehle wurde in der Vorlesung um sogenannte register-only Befehle erweitert. Ist es möglich mit den vorhandenen Datenpfaden den Befehl

`ADD ACC IN1`

zu implementieren? Und falls ja, welche Treiber müssen in der Execute-Phase aktiviert werden? Begründen Sie ihre Antwort.

- d) Sowohl auf dem linken, als auch auf dem rechten Operanden-Bus existiert die Möglichkeit eine  $\beta$  auf den Bus zu schreiben. Nennen Sie einen Befehl für den bei der Ausführung eine 0 auf den linken Operanden-Bus geschrieben werden muss. Begründen Sie Ihre Antwort.

### Lösung.

- a) Der Inhalt aus ACC kommt über ACCLd und Wert i aus I über IRd in die ALU. Das Ergebnis geht über ALUAd über Adressbus in SRAM. Dort greift es über ASMd auf die Speicherzelle zu. Der Inhalt der Adresse verläuft dann über SMDd auf Datenbus und landet schließlich auf dem SP.

-0.25 SPDd fehlt

Inhalt von SP muss in SRAM geschrieben werden

- b) Es werden mindestens zwei Treiber benötigt. Der Inhalt des IN2 kann durch den Treiber IN2Dd direkt auf den Datenbus zurück, der wiederum den DDId geöffnet haben muss, um es im SP Register ablegen zu können.

- c) Es ist möglich den Befehl zu implementieren. IN1 und ACC müssen in die ALU, dafür müssen die Treiber IN1Dd und ACCLd geöffnet werden. Anschließend wird das Ergebnis über den Treiber ALUDId in den ACC geladen.

-0.25

2 Treiber fehlen,  
DRd und ALUDId

-1 keine d)

## Aufgabe 2.

- a) Nehmen Sie an, dass  $SP$  auf die erste Speicherzelle oberhalb des Stacks zeigt, d.h. wenn der aktuelle Stack in  $M[n], \dots, M[n+1]$  abgelegt ist, dann gilt  $\langle SP \rangle = n-1$ . Diese Eigenschaft soll immer erhalten bleiben. Geben Sie eine Implementierung unter Benutzung des erweiterten  $ReTI$  Befehlssatzes aus der Vorlesung für die Stack-Operationen  $push()$  und  $pop()$  an. Hierbei soll bei  $push()$  der Inhalt des Akkumulators  $ACC$  auf den Stack gelegt werden und bei  $pop()$  der oberste Eintrag des Stack in den Akkumulator geschrieben werden.
- b) Warum mussten für die Interrupt-Behandlung die beiden atomaren Befehle  $INT\ i$  und  $RTI$  eingeführt werden? Weshalb können diese nicht einfach durch den restlichen Befehlssatz implementiert werden?

## Lösung.

- a)  $push(): STORE\ ACC\ SP$   
 $pop(): LOADIN\ SP\ ACC\ 1$

-0.25 Befehl nicht  
ihr müsst aber Platz auf dem Stack schaffen bzw. wegnehmen durch bewegen des SP-Registers.

-2

- b)  $syscall$  und  $RTI$  sind Jump-Befehle welche bewerkstelligen, dass wenn eine Interrupt-ServiceRoutine ausgeführt wird, man an die Stelle  $IVT[i]$  zurückkehren kann, indem diese auf dem Stack gespeichert und später wieder in den PC geladen und vom Stack entfernt wird. Ohne diese kann nicht wieder in den Benutzermodus gewechselt werden.

## Aufgabe 3.

beste Antwort bisher, aber die  $RETI$  hat sowas wie Benutzermodus usw. nicht, nur andere Architekturen

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

...

Interpretieren Sie die Ausgabe beginnend mit der Zeile mit dem Inhalt " $brk(0x\dots)$ ". Geben Sie an, welche Bibliotheksfunktionen für die einzelnen Systemaufrufe verantwortlich sein könnten.

## Lösung.

- 00:  $brk(0x\dots)$
- 01:  $fopen$  - öffnet Datei im Schreibmodus
- 02:  $(fstat)$
- 03:  $fprintf$
- 04:  $fprintf$
- 05:  $fprintf - i > 2500$
- 06:  $fclose$
- 07:  $fopen$  - öffnet Datei im Lesemodus
- 08:  $(fstat)$
- 09:  $fscanf$
- 10:  $fscanf$
- 11:  $fscanf$
- 12:  $fscanf - j > 2500$
- 13:  $fclose$