Betriebssysteme Übungsblatt02

Aufgabe 1

a) STOREIN ACC SP i
→ M(<ACC> + [i]) := SP
ACC geht über ACCLD in ALU und I geht über IRd auch in ALU und dann über ALUAd auf dem A Bus in SRAM. Die 2 gehen in den ALU, um das Register zum speichern zu berechnen also auf welchem Register es gespeichert werden soll. SP geht über SPDd auf D Bus in SRAM und da wird der Wert in das im ALU berechnete Register gespeichert.

b) MOVE IN2 ACC
→ ACC := IN2
Der IN2Dd Treiber wird aktiviert und der DDId Treiber. Dann kann der Wert direkt in ACC geladen werden.

c) ADD ACC IN1
→ ACC = ACC + IN1
Für den ACC wird der ACCLd Treiber aktviert und der Wert geht über den L Bus in ALU. Für das IN1 Register wird der IN1Dd Treiber aktivert für den Bus und über den aktivierten DRd Treiber auf den R Bus in den ALU. Der berechnete Wert geht dann über ALUDId wieder in ACC.

d) LOADI ACC i
Bei diesem Befehl wird eine Konstante i in den ACC geladen. Die Konstante kommt von I (IRd) und da diese nicht verändert werden darf in der Berechnung im ALU kommt vom L Bus eine Konstante 0^32 (0Ld). Dann kann die Konstante von I in ACC geladen werden (ALUDId).

Aufgabe 2
a)
LIFO-Prinzip.

- push() → Legt den Wert auf den Stack (quasi oben drauf)
  → STOREIN SP ACC 0 M(SP + 0) := ACC
  → SUBI SP 1 Stack Pointer um 1 verringern

- pop() → Holt sich den obersten Eintrag vom Stack und lädt ihn in ACC
  → LOADIN SP ACC 1;
  → ADDI SP 1;

b) Anders als bei den bisherig zur Verfügung stehenden Befehlen, ist es möglich mit den Interrupt-Befehlen den Modus zum System- oder Usermodus zu ändern. Zugriff auf die Interruptvektortabelle, auf welche so nicht über die Standard-Befehlen zugegriffen werden kann. Benötigt einen Pointer für die Adresse der Interruptvektortabelle im SRAM.

beste Antwort, die man geben kann, so wie die Aufgabe gestellt ist. Aber die RETI hat keinen Usermodus usw., aber andere Architekturen haben derartige Sachen

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

Aufgabe 3

Ausgabe interpretieren:

- brk(.....)
  Änderung des Ortes des program break, der das Ende des Datensegments des Prozesses definiert.
- openat(...)
  Öffnet bzw. erstellt eine Datei. Dieser Systemaufruf kommt durch die Bibliotheksfunktion "fopen("myfile.txt","w")" Zustande. "w" bedeutet das eine neue Datei erstellt wird mit dem Namen „myfile.txt“.
- fstat(...)
  Gibt Informationen einer Datei zurück.
- write(...)
  In einen Dateideskriptor schreiben. Dieser Systemaufruf kommt durch die Bibliotheksfunktion "fprintf(fptr,"%d ",i)" zustande, welche in eine Datei schreibt.
- read(...)
  Liest eine Datei. Dieser Systemaufruf kommt durch die Bibliotheksfunktion "fscanf(fptr, "%d ", &value)", welche eine Datei liest.
- close(...)
  Schließt eine Datei. Dieser Systemaufruf kommt durch die Bibliotheksfunktion "fclose(fptr) zustande, welche dazu dient eine geöffnete Datei zu schließen.
- exit_group(42)
  Alle Threads in einem Prozess beenden