Digitale Systeme SS2019

Übungsblatt 2

Aufgabe 1 **(3 Punkte)**

Notieren Sie die nachfolgend angegebenen Maschinenbefehle eines Prozessors (Beschreibung in „mproz.pdf“) in Assembler-Mnemonic!

Unterlagen zum Prozessor finden Sie im Download-Bereich der Übungen.

(a) `0001000010000100`

(b) `0110010111000011`
`0000000000010001`

(c) 0000110101010010

Aufgabe 2 **(3 Punkte)**

Entwerfen Sie ein Befehlsformat für einen Prozessor unter Beachtung folgender Angaben:

- 16 Befehle
- 2 - Adressmaschine
- Wortbreite 16 bit
- 16 Register
- 16 MWorte Hauptspeicher sind adressierbar
- Speicher soll byte-adressierbar sein
- Konstanten sind 1 Wort breit
- Adressierungsarten: registerdirekt, registerindirekt, speicherdirekt, unmittelbar, registerindiziert mit 16 bit Index

Einer der beiden Operanden ist ausschließlich registerdirekt adressiert. Die Richtung des Datentransfers ist festzulegen.

Die Befehle seien ganze Vielfache von Worten lang.

Aufgabe 3 **(4 Punkte)**

Nehmen Sie eine interne Prozessorarchitektur entsprechend untenstehender Grafik an.

Gegeben seien das folgenden, in Assembler-Mnemonik dargestellte, Programmfragment:

```
SUB [R1], 8[-R2]
JRZ 42
```

Erster Befehl: `"SUB op1, op2": Subtraktion: op1 = op1 - op2`; der erste Operand wird registerindirekt adressiert (`"[Rx]"`), der zweite registerindiziert mit Predekrement (`"n[-Rx]"`). Der Befehl ist 1 Wort lang

Zweiter Befehl: `"JRZ Sprungdistanz"`, wenn das Zero-Flag gesetzt ist, Sprung zur Programmspeicheradresse `"IP + Sprungdistanz"` (relativer Sprung); der Befehl ist 2 Worte lang (das 2. Wort ist die Sprungdistanz)

Y ist ein Hilfsregister, das im Mikroprogramm verwendet werden kann. R1 und R2 sind Anwendungsregister, die vom Mikroprogramm nur in der im Maschinenbefehl spezifizierten Weise verändert werden dürfen.

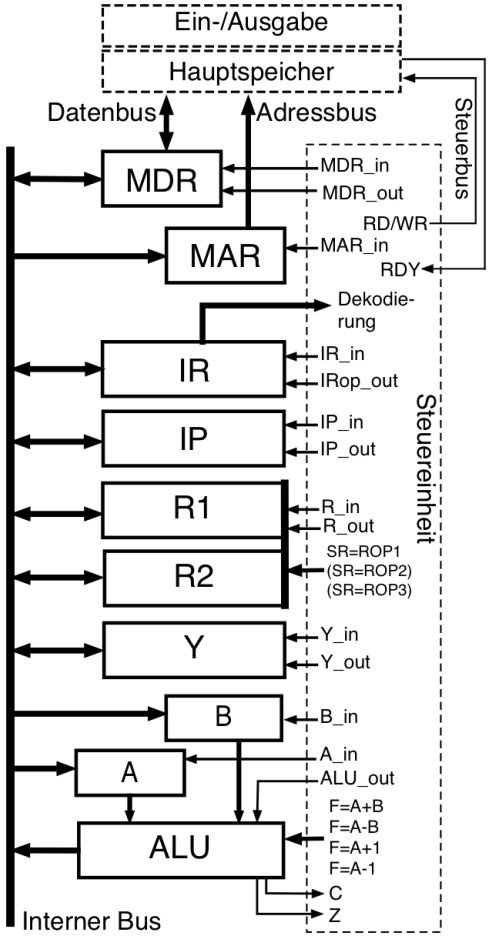
Eine gesetzte Funktion der ALU (`F=...`) bleibt bis zum nächsten Setzen erhalten.

Die Inhalte der Register `MAR`, sowie `MDR` bei Schreibzugriffen, müssen während des gesamten Speicherzugriffs erhalten bleiben.

Die folgenden Signale können in Mikrobefehlen zusätzlich verwendet werden:

- `IRop_out` liest den Bereich des Befehlsregisters (IR), in welchem, bei indizierter Adressierung, der Index enthalten ist
- `DECODE`: löst die Dekodierung des Inhalts des Registers IR und eine entsprechende Verzweigung zu einer der angegebenen Adressen im Mikroprogramm aus.
  Notation:
  `DECODE['Symbol für Opcode der Instruktion':'Adresse im Mikroprogramm', ...]`
  Beispiel: `DECODE[mov:3,add:17,jpz:10]`
- `WRDY` : die Ausführung des nächsten Mikrobefehls wird verzögert, bis der Speicherzugriff beendet ist
- `RD` : nach Ausführung des Mikrobefehls wird dem externen Speicher ein Lesevorgang signalisiert; das Signal braucht nur einmalig aktiv zu werden und wirkt bis zum Abschluss des Speicherzugriffs
- `WR` : nach Ausführung des Mikrobefehls wird dem externen Speicher ein Schreibvorgang signalisiert; das Signal braucht nur einmalig aktiv zu werden und wirkt bis zum Abschluss des Speicherzugriffs
- `SR`: wählt eines der Universalregister aus; zur Auswahl dienen die Bits für Operandenadressen (`ROPx`)
- `R_in`, `R_out` : liest bzw. schreibt den Inhalt eines der im Maschinenbefehl angegebenen und durch `SR` ausgewählten Register

- `FLE` : erlaubt das Setzen der ALU-Flags durch die ausgeführte ALU-Operation (bei Verwendung der ALU für Adressberechnungen sollten die ALU-Flags nicht verändert werden)
- `IFZ` : testet den Zustand des Zero-Flags: wenn es gesetzt ist, wird der Mikrobefehl an der angegebenen Adresse ausgeführt. Wenn nicht, wird der Mikrobefehl an Adresse 0 ausgeführt. (Bsp.: `IFZ[61]`)
- `NA` : eine Gruppe von Signalen, welche die Adresse des nächsten Mikrobefehls darstellt (entspricht einem Sprung im Mikroprogramm, z.B. `NA[5]`)



Schreiben Sie ein Mikroprogramm welches einen Prozessor in die Lage versetzt, die oben genannten Befehle ausführen zu können. Verwenden Sie die nachstehende Tabelle.

- das Mikroprogramm bestehe aus 3 Teilen, die nacheinander im Mikroprogrammspeicher angeordnet sein sollen: 1. Befehl holen, 2. Ausführung von JRZ, 3. Ausführung von SUB
- machen Sie diese Teile in Ihrem Mikroprogramm kenntlich
- vermeiden Sie unnötig lange Wartezeiten bei Speicherzugriffen

Adresse	Mikrobefehl	Kommentar
0		