Wiederholungsklausur

Einführung in die Rechnerarchitektur

Prof. Dr. Arndt Bode

Sommersemester 2015
8. April 2015

Name: _____

Vorname: _____

Matrikelnummer: _____

Geburtsdatum: _____

Hörsaal: _____

Platz: _____

Unterschrift: _____

Ergebnis:

Aufgabe	1	2	3	4	Ges.	Note
Punkte						
Korrektur						

Hinweise zu den Aufgaben

- Die Bearbeitungszeit beträgt 120 Minuten.
- Es sind keinerlei Hilfsmittel zugelassen, auch keine Taschenrechner.
- Versehen Sie dieses Angabenblatt auf der Titelseite mit Ihrem Namen, Vornamen und Matrikelnummer, sowie Geburtsdatum, Hörsaal und Platznummer. Vergessen Sie nicht die Unterschrift!
- Diese Angabe umfasst *28* bedruckte Seiten (inklusive Deckblatt).
- Außerdem erhalten Sie die folgenden Merkblätter:

  Anlage I: „Die wichtigsten 80386 Befehle und ihre Operanden“

  Anlage II: „Mikroprogrammierung“

  Anlage III: „Die wichtigsten VHDL Konstrukte“

- Alle Lösungen sind in dieses Heft einzutragen. Sollte der vorgesehene Platz nicht ausreichen, so finden Sie am Ende weitere Blätter. Sollten auch diese nicht ausreichen, wenden Sie sich bitte an die Aufsichten.
- Notizpapier wird auf Ihre Anfrage ausgegeben. Die Verwendung von eigenem Papier ist **nicht** gestattet.

*Lösungen auf Notizpapier werden **nicht** gewertet!*

Aufgabe 1 - Maschinennahe Programmierung

1.1 Einzelbefehle

1.1.1 Vereinfachung

Fassen Sie die folgenden Befehle zu einem Befehl mit dem gleichen Resultat zusammen.

```
sub esp,4
mov [esp],eax
```

1.1.2 Befehlssequenzen

Implementieren Sie kurze Befehlssequenzen, um die genannten Effekte zu bewirken. Temporär genutzte Register müssen nicht gesichert werden.

- BX ins höherwertige Wort von EAX und CX ins niederwertige Wort von EAX kopieren
- Invertieren der Bits 0 und 8 im Register AX

1.2 Programmverständnis

1.2.1 Statusregister

Kreuzen Sie die Status-Flags an, welche nach Ausführung eines jeden Befehls gesetzt sind. Jeder Befehl ist unabhängig zu betrachten; vor jedem Aufruf sind alle Status-Flags nicht gesetzt. Im Register EAX steht vor der Ausführung des Befehls der Wert „300“.

Befehlsfolge	Carry	Zero	Sign	Overflow
sub al,100	____	____	____	____
mov ax,300	____	____	____	____
cmp ah,200	____	____	____	____
add al,100	____	____	____	____

1.2.2 Registerbelegung

Geben Sie die Inhalte der gefragten Register (in Hex oder Dec) nach der Ausführung des folgenden Programms an. Der Stapelspeicher ist zu Beginn leer (ESP=„4096“), die Werte der frei belegbaren Register alle „0“.

```
variable:
  db 42
array:
  db 23,47,11
table:
  dw 10,20,30,40

mov eax,[variable]
push eax
mov ax,[array]
mov ebx,table
mov ecx,[ebx]
mov edx,[ebx+1]
pop bx
```

Registerwerte nach Programmausführung

AH _____ BL _____ CX _____ DH _____ ESP _____

1.3 Programmentwicklung

- Unterprogramme sollen neben den in der Aufgabenstellung explizit genannten keine weiteren Seiteneffekte haben.
- In der Aufgabenstellung nicht erwähnte Sonderfälle müssen bei der Bearbeitung nicht berücksichtigt werden.
- Zu nutzende Speicherbereiche dürfen als hinreichend groß und sich nicht überschneidend angenommen werden.

1.3.1 Absolutbetrag

Schreiben Sie ein per `CALL` aufrufbares Unterprogramm `absolut`, welches den Absolutbetrag einer vorzeichenbehafteten Zahl im Register `EAX` bildet und in selbigem zurückgibt.

1.3.2 Befehlsnachbildung

Der `MOVSX`-Befehl kopiert ein vorzeichenbehaftetes Datum an ein Ziel größerer Breite und füllt die zusätzlichen Bits mit dem Vorzeichenbit des kopierten Datums auf. Schreiben Sie ein per `CALL` aufrufbares Unterprogramm `mymovsx`, welches diese Funktionalität nachbildet, indem es ein in `DX` übergebenes Datum vorzeichen-erweitert in `EAX` zurückgibt.

1.3.3 Formatumwandlung

- Eine C-Zeichenkette ist eine Folge unbestimmter Länge von Bytes (ASCII-Zeichen), welche durch ein `0`-Byte terminiert wird.
- Eine Pascal-Zeichenkette ist eine Folge von Bytes (ASCII-Zeichen), deren erstes Byte die Anzahl der anschließenden relevanten Bytes als vorzeichenlose Zahl angibt.

Schreiben Sie ein per `CALL` aufrufbares Unterprogramm `c2p`, welches eine C-Zeichenkette (max. 255 Zeichen garantiert) an der in `EAX` gespeicherten Arbeitsspeicheradresse unmittelbar in eine Pascal-Zeichenkette an die in `EBX` übergebenen Adresse konvertiert. Als Rückgabewert soll in `EAX` die Länge der Zeichenkette übergeben werden.

Aufgabe 2 - Mikroprogrammierung

Sie haben ein Merkblatt mit einer Kurzbeschreibung der mikroprogrammierbaren Maschine erhalten, in welchem Sie alle zur Lösung der Aufgabe notwendigen Angaben finden, z.B. die Beschreibung des Mikroinstruktionsformats und die Funktionstabellen der Bausteine.

Das Mikroprogramm IFETCH, abgelegt ab Adresse 0x000 des Mikroprogrammspeichers, kann als gegeben betrachtet werden. Es holt den nächsten Maschinenbefehl aus dem Hauptspeicher in das Instruktionsregister, inkrementiert den Befehlszähler und springt das dem Befehls-Opcode zugehörige Mikroprogramm an. IFETCH benötigt 3 Takte zur Ausführung. Mikroprogramme müssen am Ende wieder zum Anfang des Mikroprogramms IFETCH zurückspringen. Hexadezimale Zahlen werden in Java- bzw. C-Schreibweise dargestellt und beginnen mit der Buchstabenkombination 0x, z.B: 0x1234, 0x011A oder 0xBEEF.

Gegeben sind folgende Maschinenbefehle:

Opc.	Befehl		Beschreibung
0x05	`MOV`	[**RA**],**RB**	Kopiert den Inhalt der Speicherstelle, deren Adresse in Register **RA** steht, in das Register **RB**.
0x11	`INC`	**RB**	Inkrementiert **RB** um 1 (**RB** = **RB** + 1). Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt.
0x13	`INC`	[**RB**]	Inkrementiert den Inhalt der Speicherstelle, deren Adresse in Register **RB** steht, um 1 ([**RB**] = [**RB**] + 1). Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt.
0x19	`DEC`	**RB**	Dekrementiert **RB** um 1 (**RB** = **RB** - 1). Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt.
0x22	`ADD`	**RA**, **RB**	Addiert den Inhalt von Register **RA** zu Register **RB**.
0x32	`CMP`	**RA**, *imm*	Führt eine Subtraktion des Inhalts von **RA** minus *imm* durch. Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt. Das Ergebnis wird verworfen.
0x40	`JZ`	*adr*	Führt einen Sprung an die Adresse *adr* aus, wenn das Zero-Flag des Maschinenstatusregisters gesetzt ist.
0x41	`JNZ`	*adr*	Führt einen Sprung an die Adresse *adr* aus, wenn das Zero-Flag des Maschinenstatusregisters nicht gesetzt ist.
0x42	`JGE`	*adr*	Führt einen Sprung an die Adresse *adr* aus, wenn das Carry-Flag des Maschinenstatusregisters gesetzt ist.

Außerdem ist folgender komplexerer Maschinenbefehl gegeben:

Opc.	Befehl		Beschreibung
0x59	`INCI`	[**RA** + [**RB**]], *imm*	Bedingtes indiziertes Inkrement: Ermittelt einen Index **I** als Inhalt der Speicherstelle, deren Adresse in Register **RB** gegeben ist. Nur wenn dieser Index **I** kleiner als *imm* ist, erfolgt eine Inkrementierung des Inhalts der Speicherstelle, deren Adresse durch **RA + I** gegeben ist, um 1. Das Maschinenstatusregister wird nicht beeinflußt.

Anmerkung: Die Bezeichnung „**RA**“ (bzw. „**RB**“) steht hier abkürzend für „das durch das A-Registeradressfeld (bzw. B-Registeradressfeld) des Maschinenbefehlswortes adressierte Register“.

2.1 Welche der oben angegebenen Befehle belegen

a) 16 Bit

---

b) 32 Bit

---

im Hauptspeicher? Bitte die jeweiligen Opcodes angeben!

2.2 Der Opcode `JGE` bedeutet „jump if greater equal“. Begründen Sie kurz, warum dies für die MI-Maschine mit der Bedingung „C-Flag gesetzt“ gleichgesetzt werden kann.

2.3 Gegeben ist folgendes Maschinenprogramm, das ein Datenfeld abgelegt an einer Startadresse (in Register r0) mit einer Länge (in Register r1) verarbeitet. Weitere Eingabe ist r2 als Startadresse eines Ausgabefeldes.

```
            CMP  r1, 0
            JZ   ende
schleife:   MOV  [r0], r3
            CMP  r3, 4
            JGE  weiter
            ADD  r2, r3
            INC  [r3]
weiter:     INC  r0
            DEC  r1
            JNZ  schleife
ende:       ...
```

☐ **2.3.1** Die Inhalte der Speicherstellen 0x0100 bis 0x0104 seien mit 0 vorbelegt, die von 0x0200 bis 0x0204 mit 2,3,4,3,2. Wenn man das gegebene Maschinenprogramm mit r0 = 0x0200, r1 = 5, r2 = 0x0100 aufruft, welche Inhalte stehen nach Ablauf in den Speicherstellen 0x0100 bis 0x0104?

Addresse	0x0200	0x0201	0x0202	0x0203	0x0204
Wert	2	3	4	3	2

Addresse	0x0100	0x0101	0x0102	0x0103	0x0104
Wert					

☐ **2.3.2** Beschreiben Sie die Aufgabe des Programms.

☐ **2.3.3** Die 5 Maschinenbefehle ab *schleife* bis *weiter* können durch `INCI` [r2 + [r0]], 4 ersetzt werden. Wie viele Speicherzugriffe finden mit $[r0] = 0$ in diesem Bereich statt – zum einen durch die ursprünglichen Maschinenbefehle, zum anderen durch `INCI`? Begründen Sie Ihre Lösung, und berücksichtigen Sie `IFETCH`.

☐ **2.3.4** Zeigen Sie, wie das gegebene Maschinenprogramm in hexadezimaler Codierung aussieht.

Adresse	Inhalt	Befehl		
0x0100	_____		`CMP`	r1, 0
_____	_____			
_____	_____		`JZ`	*ende*
_____	_____			
_____	_____	*schleife*:	`MOV`	[r0], r3
_____	_____			
_____	_____		`CMP`	r3, 4
_____	_____			
_____	_____		`JGE`	*weiter*
_____	_____			
_____	_____		`ADD`	r2, r3
_____	_____			
_____	_____		`INC`	[r3]
_____	_____			
_____	_____	*weiter*:	`INC`	r0
_____	_____			
_____	_____		`DEC`	r1
_____	_____			
_____	_____		`JNZ`	*schleife*
_____	_____			
_____		*ende*:	...	

2.4 Folgende Maschinenbefehle sollen nun durch ein Mikroprogramm realisiert werden:

- `CMP` **RA**, *imm*
- `INCI` [**RA**+[**RB**]],*imm*

☐ **2.4.1** Vervollständigen Sie die Tabellen auf den Seiten 18/19.
Die Tabelle enthält Reservezeilen für einen zweiten Lösungsversuch, bitte streichen Sie falsche Lösungen deutlich durch!
Bitte keine binären Werte eintragen, sondern die Abkürzungen aus dem Merkblatt MI (in der Anlage) verwenden.

79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39
IE	I3	I2	I1	I0	KMUX	K15	K14	K13	K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1	K0	I2	I1	I0	I5	I4	I3	I8	I7	I6	A3	A2	A1	A0	ASEL	B3	B2	B1	B0	BSEL
	Interrupt					Konstante																Src			Func			Dest			RA Addr					RB Addr				

Adr. CMP RA,imm

	DIS	X		X							
	DIS	X		X							

INCI [RA+[RB]],imm

	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							

Adr. CMP RA,imm

	DIS	X		X							
	DIS	X		X							

INCI [RA+[RB]],imm

	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							
	DIS	X		X							

38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABUS*	DBUS*	I12	I11	I9	I8	I7	I6	CEMUE*	CEM*	I5	I4	I3	I2	I1	I0	CCEN*	I3	I2	I1	I0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	BZ_LD*	BZ_ED*	BZ_INC*	BZ_EA*	IR_LD*	MWE*
Y-MUX		CIN MUX		Schiebe-steuerung						Statusregister Test							AM2910 Befehle				Direktdaten																			

			X	H	H	X																																
			X																																			

			X	H	H	X																																
			X	H	H	X																																
			X	H	H	X																												E				
			X																																			
			X	H	H	X																																
			X																																			
			X	H	H	X																																
			X	H	H	X																																

			X	H	H	X																																
			X																																			

			X	H	H	X																																
			X	H	H	X																																
			X	H	H	X																												E				
			X																																			
			X	H	H	X																																
			X																																			
			X	H	H	X																																
			X	H	H	X																																

Aufgabe 3 - Rechnergestützter Schaltungsentwurf

3.1 Faktorisierung

Es soll eine logische Schaltung erstellt werden, die die Anzahl der Teiler einer Zahl berechnet. Sowohl die Eingänge $a_2a_1a_0$ als auch die Ausgänge $b_2b_1b_0$ werden dabei als dreibittige vorzeichenlose Zahlen interpretiert. Um den Spezialfall $a_2a_1a_0 = 000$ sinnvoll zu definieren, wird dieser Eingangskombination abweichend von der normalen Binärkodierung der Zahlenwert 8 zugeordnet. Beispiel: Die Zahl 6 hat die Teiler 1, 2, 3 und 6, also ist der gesuchte Ausgangswert $b_2b_1b_0$ gleich 100.

☐ **3.1.1** Ergänzen Sie die Wertetabelle der Schaltung!

a_2	a_1	a_0	b_2	b_1	b_0
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0	1	0	0
1	1	1			

☐ **3.1.2** Zeichnen Sie ein Logikdiagramm einer Schaltung, die b_2 bis b_0 als Eingang bekommt und deren einziger Ausgang c den Wert 1 hat, wenn $a_2a_1a_0$ eine Primzahl ist!

☐ **3.1.3** Geben Sie eine disjunktive Normalform (DNF) für b_1 an!

☐ **3.1.4** Geben Sie eine konjunktive Normalform (KNF) für b_0 an!

☐ **3.1.5** Erstellen Sie eine VHDL-Architecture, die die Ausgänge entsprechend der Wertetabelle berechnet!

Die folgende VHDL-Entity-Deklaration ist gegeben:

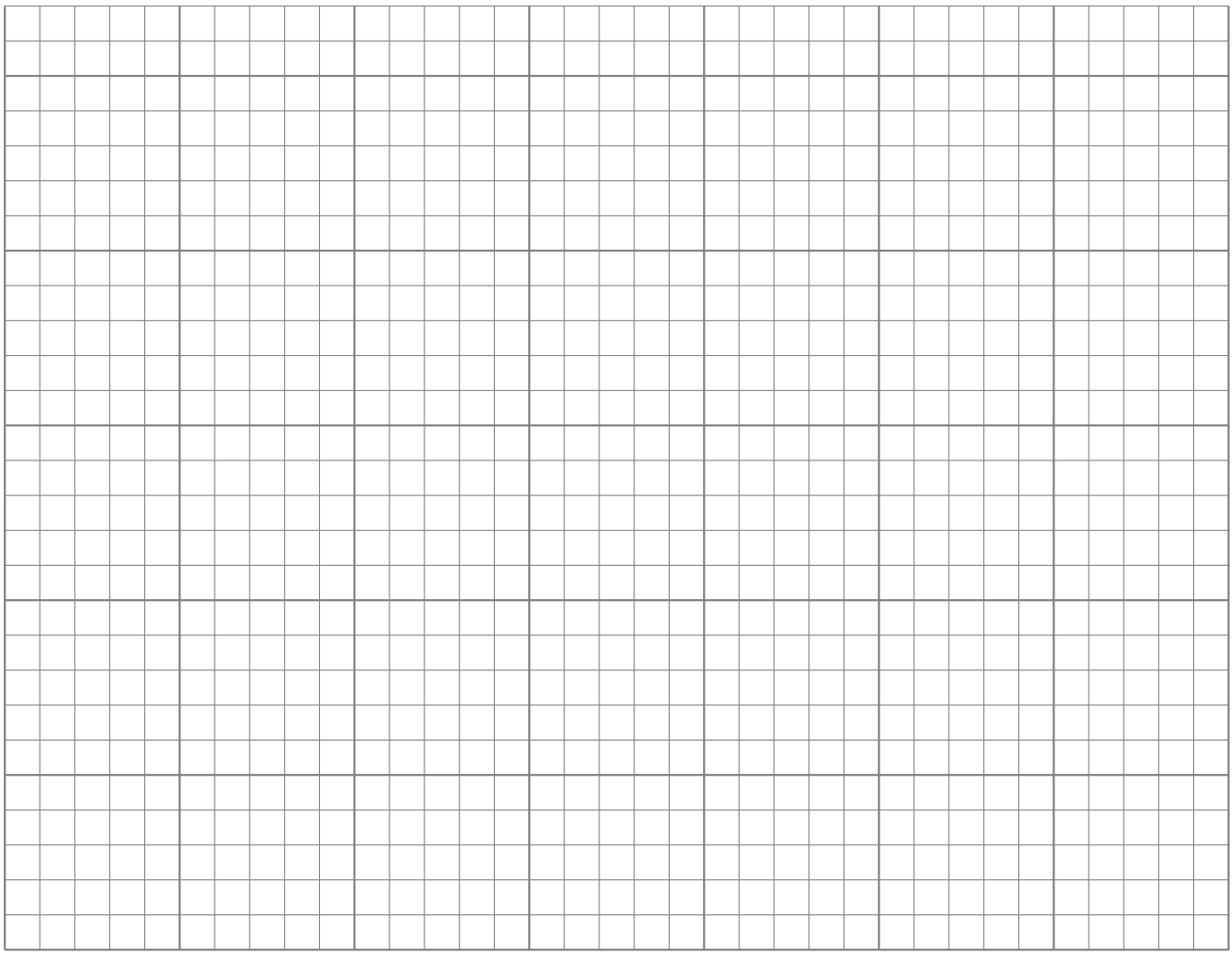
```
entity factors is
    port(
        a: in  unsigned(2 downto 0);
        b: out unsigned(2 downto 0)
    );
end entity;
```

3.2 Endlicher Automat

Ein endlicher Automat ist wie folgt beschrieben:

- Der Automat hat die Zustände z_0, z_1, z_2, z_3 und z_4.
- Zustandsübergänge erfolgen nur bei einer steigenden Flanke des Eingangs `clk`.
- Ist der Eingang `start` gesetzt, so ist der nächste Zustand immer der Startzustand z_0 (synchroner Reset).
- Sind im Zustand z_i mit $0 \leq i < 5$ beide Eingänge `start` und `input` nicht gesetzt, so ist der nächste Zustand $z_{2i \bmod 5}$.
- Ist nur `input` gesetzt, ist der nächste Zustand $z_{(2i+1) \bmod 5}$.
- Der Ausgang `output` ist im Zustand z_0 auf 1 gesetzt, in allen anderen Zuständen auf 0.

☐ **3.2.1** Zeichnen Sie das Zustandsübergangsdiagramm („Blasendiagramm“) des Automaten!



☐ **3.2.2** Handelt es sich um einen Moore-Automaten oder um einen Mealy-Automaten? Begründen Sie Ihre Aussage!

Der Automat ist dafür vorgesehen, vorzeichenlose Binärzahlen zu verarbeiten. Für jede Zahl a wird zuerst für einen Takt der `start`-Eingang gesetzt, anschließend wird die Zahl seriell auf den Eingang `input` gelegt. Dazu wird mit dem höchstwertigen Bit von a begonnen. In dem Takt nach Eingabe des niederwertigsten Bits von a wird der Ausgang `output` evaluiert.

☐ **3.2.3** Bestimmen Sie für jeden der angegebenen Werte von a den Wert von `output`, nachdem die Zahl wie oben beschrieben an den Automaten übergeben wurde!

a (dezimal)	a (binär)	`output`
4	100	
5	101	
10	1010	
20	10100	
6	110	
11	1011	
25	11001	
50	110010	
100	1100100	

☐ **3.2.4** Welche Funktion erfüllt der Automat?

☐ **3.2.5** Erstellen Sie die VHDL-Entity des Bausteins! Halten Sie sich an die vorgegebenen Signalnamen.

☐ **3.2.6** Erstellen Sie die VHDL-Architecture des Bausteins!

Zusätzlicher Platz für Lösungen - Bitte immer Aufgabennummer angeben!

Lösung für Aufgabe . . .

Lösung für Aufgabe ...

Lösung für Aufgabe ...

Lösung für Aufgabe ...

Lösung für Aufgabe ...

Lösung für Aufgabe ...

Lösung für Aufgabe ...