Technische Hochschule Ingolstadt - Fakultät Informatik
Prof. Dr. Nikoletta Sofra- Wintersemester 2025-26

Praktikum Rechnerarchitektur - Informatik BSc - Blatt IV

Berechnung der ISBN-13 Prüfziffer

- **Verwendung einer Prozedur**
- **Linken separat assemblierter Quellcode Dateien**

Für das letzte Testat müssen Sie ein Programm in x86 Assembly schreiben, welches als Eingabe die ersten 15 Zeichen (12 numerische Dezimalziffer und 3 Bindestriche) einer ISBN entgegennimmt und die berechnete Prüfziffer (das 16. Zeichen bzw. 13e numerische Dezimalziffer der ISBN) ausgibt.

Folgende Anforderungen müssen erfüllt werden:

- Die Eingabe wird aus der Kommandozeile als erstes Nutzerargument angegeben, z.B.: `$ ./ISBN 978-3-455-01430`
- Wenn das Format der Eingabe falsch ist, dann soll das Programm eine geeignete Meldung ausgeben und beenden. Das Programm muss überprüfen, dass genau 12 numerische Ziffer und 3 Bindestriche in einem String angegeben worden sind. Dabei spielt die Position der Ziffer keine Rolle.
- Wenn das Format der Eingabe korrekt ist, dann wird die berechnete Prüfziffer gefolgt von einem Zeilenbruch ausgegeben.
- Die Berechnung der Prüfziffer soll in einer separaten Prozedur programmiert werden. Diese Prozedur enthält die Addresse des vom Nutzer eingegebenen Strings als Eingabe und liefert die berechnete Prüfziffer zurück.
- Die Prozedur soll die Adresse des eingegebenen Strings über dem Stack entgegennehmen und ihren Rückgabewert im Register **EAX** speichern. Wenn das Format der Eingabe korrekt ist, soll der Rückgabewert der Prozedur die berechnete Ziffer sein. Wenn das Format der Eingabe falsch ist, dann soll der Rückgabewert -1 sein.
- Das Hauptprogramm übernimmt das Einlesen der Adresse der Nutzereingabe, den Aufrufen der Prozedur und die Ausgabe.

Die Umsetzung folgender Anforderung ist optional:

- Die Prozedur soll in einer separaten Datei definiert und separat vom Hauptprogramm assembliert werden.

Weiterführende Information, Beispiele, Hinweise:

A. Eingabeformat

Das Eingabeformat gilt als korrekt, wenn die Eingabe genau aus 12 Dezimalziffern (0-9) und 3 Bindestrichen besteht. Leerzeichen sind nicht erlaubt. Eine Eingabe muss als falsch eingestuft werden, wenn mehr/weniger/andere Symbole als die erlaubten enthalten sind. Beispiele falscher Eingaben:

- mehr oder weniger als 12 Dezimalziffer
- mehr oder weniger als 3 Bindestriche
- Verwendung anderer Zeichen (z.B. Buchstaben) im Eingabestring
- Runterbrechen der Eingabe in mehreren Strings.

B. ISBN (International Standard Book Number) – 13

Quelle: ISBN Users' Manual, Seventh Edition

Die Prüfziffer ISBN dient der Erkennung von Fehlern bei der ISBN-Eingabe und wird wie folgt gerechnet:

- Die ersten zwölf Dezimalziffer der ISBN werden abwechselnd mit den Faktoren 1 und 3 multipliziert. Die resultierenden Produkte werden aufsummiert. (Anders formuliert, es wird eine gewichtete Stumme aller 12 Dezimalziffer erstellt, bei der jede zweite Ziffer mit 3 multipliziert wird).
- Die Prüfziffer ergibt sich aus der Zahl, die wir zu der gewichteten Summe addieren müssen, damit diese ein Vielfaches vom 10 wird.
    - Spezialfall: wenn die Gewichtssumme ein vielfaches von 10 ist, dann ist die Prüfziffer 0
    - Sonst muss die Gewichtssumme mit 10 dividiert werden. Der Quotient der Division ist egal. Wir verwenden den Divisionsrest um die Prüfziffer zu finden: Dir Prüfziffer ist das Ergebnis der Subtraktion des Rests von der Zahl 10.
- **Zusammengefasst:**
  Prüfziffer = (10 – (gewichtete Summe mod 10)) mod 10

  Beispiel:

	Präfix			Gruppen Nr.	Verlags-Nr.			Titelnummer					Gewichtete Summe
ISBN	9	7	8	1	2	9	2	4	2	0	1	0	
Gewicht	1	3	1	3	1	3	1	3	1	3	1	3	
Produkt	9	21	8	3	2	27	2	12	2	0	1	0	87

Prüfziffer = 10 – ((gewichtete Summe) mod 10) =

= 10 – (87 mod 10) = 10 -7 = 3

Weitere Beispiele:

978-3-8458-3851 : Prüfziffer = 9

978-3-455-01430 : Prüfziffer = 3

978-3-518-46920 : Prüfziffer = 0

C. Ablage von Strings im Speicher

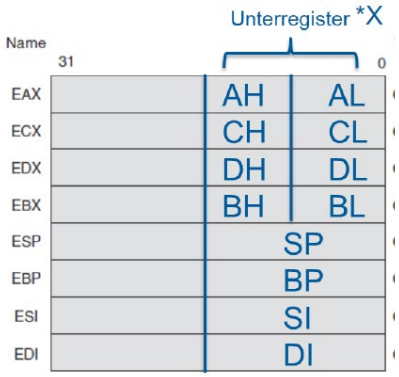
```
(gdb) x /3s 0xbffff3aa
0xbffff3aa:     "Computer"
0xbffff3b3:     "Architecture"
0xbffff3c0:     "Rules"
(gdb) x /28c 0xbffff3aa
0xbffff3aa:     67 'C'  111 'o' 109 'm' 112 'p' 117 'u' 116 't' 101 'e' 114 'r'
0xbffff3b2:     0 '\000'        65 'A'  114 'r' 99 'c'  104 'h' 105 'i' 116 't' 101 'e'
0xbffff3ba:     99 'c'  116 't' 117 'u' 114 'r' 101 'e' 0 '\000'        82 'R'  117 'u'
0xbffff3c2:     108 'l' 101 'e' 115 's' 0 '\000'
(gdb)
```

lars@ws23-AsmCpp2...

Weil b2-aa = 8 😊 sehen wir, dass jeder Character des Strings ein Byte im Speicher besetzt. Wir sehen auch, dass jeder String mit dem Wert NUL - 0 terminiert wird.

D. Auffrischung: Register

Die Teile der Register (Unterregister) rechts können als eigenständige Register genutzt werden, z.B. AX ist 2 Byte lang, AH und AL jeweils 1 Byte.



E. Die Instruktion `div`

Die Operation ist Division (nicht vorzeichenbehaftet). Die Funktion div verwendet implizite Adressierung, das heißt einige Operanden sind im Opcode mitenthalten. Wir können nur den Divisor als Operand angeben. Der Dividend und die Zielregister (Ablage Quotient und Division) sind abhängig von der Bitlänge des Divisors vordefiniert:

Wenn der Operand (Divisor) 1 Byte lang ist, wird der Inhalt von AX durch den Divisor geteilt. Das Ergebnis wird wie folgt abgelegt:

- Quotient in AL
- Rest in AH.

Beispiel:
`div bl;` resultiert in AL ← AX/BL, AH ← AX%BL

Wenn der Operand (Divisor) 2 Bytes lang ist, wird der Inhalt von DX:AX durch den Divisor geteilt. *DX:AX heißt, dass die jeweils 16-Bit von DX und AX zu einem 32-Bit Wert konkateniert werden und als ein einzelner Operand verwenden werden.*
Das Ergebnis wird wie folgt abgelegt:

- Quotient in AX
- Rest in DX.

Beispiel:
`div bx;` resultiert in AX ← DX:AX/BX, DX ← DX:AX%BX

Wenn der Operand (Divisor) 4 Bytes lang ist, wird der Inhalt von EDX:EAX durch den Divisor geteilt.
Das Ergebnis wird wie folgt abgelegt:

- Quotient in EAX
- Rest in EDX.

Beispiel:
`div ebx;` resultiert in EAX ← EDX:EAX/EBX, EDX ← EDX:EAX%EBX

F. Auffrischung: Prozeduren

Siehe auch Foliensatz Befehlssatzarchitekturen Folien 68-69

Der Aufruf einer Prozedur erfolgt durch einen **call**-Befehl. Als Argument für den **call**-Befehl verwenden Sie das Label der ersten Instruktion der Prozedur. Mit diesem Aufruf verzweigt der Kontrollfluss in die Prozedur.

Die Prozedur endet mit dem Rücksprungbefehl **ret.** Dieser bewirkt, dass die Ausführung des übergeordneten Programms mit demjenigen Befehl fortgesetzt wird, der dem call-Befehl folgt.

Sie müssen folgende Konvention beachten:

- Parameter werden der Prozedur über dem Stack übergeben – Sie müssen also den Input für Ihr Unterprogramm auf dem Stack schieben, so wie Sie das z.B. vom call printf machen.
- Wie auch oben beschrieben, der Rückgabewert soll über das Register EAX zurückgegeben werden
- Die Register EAX, ECX, EDX dürfen von der Prozedur geändert werden, das Hauptprogramm muss die ggf. sichern
  Alle anderen Register müssen von Ihrer Prozedur so verlassen werden, wie die vor dem Call waren. Die müssen also ggf. vor Benutzung gesichert und nach Benutzung bzw. vor Rückgabe zum ursprünglichen Wert zurückgesetzt werden

Optional: Lokale Variablen der Prozedur können auch temporär auf dem Stack gespeichert werden (Subtraktion benötigter Anzahl Bytes von esp, siehe Folie 69).

G. Auffrischung ASCII Zeichen:

0	NUL	16	DLE	32	SP	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	TAB	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

Hinweise:
- Die Dezimalziffer werden als Werte 48 -57 dargestellt.
- Um die Dezimalzahl einer Ziffer zu bekommen, müssen wir vom gespeicherten ASCII-Wert den Wert von 0 abziehen: ´0´= 48 = 0x30
- Der Bindestrich wird als 45 = 0x2D dargestellt.

H. Assemblieren und Linken separater Dateien (optionale Anforderung)

Symbole, die in einer Datei definiert werden und woanders genutzt werden sollen, müssen deklariert werden:

- **Extern** importiert Symbole aus anderen Objektmodulen
- **Global** exportiert Symbole in andere Objektmodule

Die Auflösung übernimmt der Linker. Zwei getrennt voneinander assemblierte Quellcode-Dateien werden zusammengebunden, z.B.:

```
> gcc lab4main.o isbnProc.o -o lab4.out
```

Für das letzte Testat müssen Sie bis Ende KW51:

Als Team ausschließlich in den Praktikumszeiten Ihrer Gruppe:

- Ihr Programm, welches alle Pflicht-anforderungen erfüllt, auf einem Laborrechner zeigen und Fragen dazu beantworten können.
- Ihr Programm assemblieren, linken und mit unterschiedlichen vorgegebenen Eingabewerten ausführen.

Das Programm (ggf. die zwei .asm Dateien) ins Moodle hochladen (oben in den Kommentaren schreiben Sie bitte mit wem Sie im Team zusammenarbeiten). Es ist wichtig, dass alle Studierende das Programm separat hochladen, damit die Testatvergabe in Moodle bestätigt werden kann!