Module 7061: Grundlagen der Informatik
Final Exam I1p 1. Februar 2013 Name: _____

Dauer: 2h30min (13h00 – 15h30)
Hilfsmittel: Erlaubt sind alle Unterlagen, Tabellen, Papier etc. Ausser den Fingern sind *keine* digitalen Hilfsmittel (Computer, Handy, Taschenrechner etc.) erlaubt. Auf dem Tisch liegen keine Tablets, ifones, smartfones und sonstiges Zeugs rum.
Lösungen: Schreiben Sie die Lösungen an die jeweils dafür reservierte Stelle auf. Bei Auswahlaufgaben kreisen Sie die entsprechenden Buchstaben ein. Zusätzliche Blätter auf denen Sie eventuell Ihren Lösungsweg aufgeschrieben haben können Sie mit abgeben. Schreiben Sie Ihren Namen hin.

Kodierungen von Werten

(5) 1. (a) Konvertiere B'A1'10'C1'C8'5A'D8$_{16}$ ins Binäre System:

B'A1'10'C1'C8'5A'D8$_{16}$ = ... 2

(b) Konvertiere 1110'0011'1010'1011'1011'1001'0101$_2$ ins Hexadezimale System:

1011'1001'1100'1101'1101'0101'0110$_2$ = ... 16

(c) Konvertiere 802$_{10}$ ins Binäre System:

802$_{10}$ = ... 2

(d) Konvertiere 101'1100$_2$ ins Dezimale System:

101'1100$_2$ = ... 10

(e) Konvertiere 122$_8$ ins Siebner-System:

122$_8$ = ... 7

(1) 2. Anzahl Speicherzellen die ein CPU mit einem Adressbus der Breite 10 ansprechen kann? .....

(1) 3. Wieviele verschiedene Werte können mit vier Bits repräsentiert werden? .....

(1) 4. Wieviele verschiedene Werte können mit einem Byte repräsentiert werden? .....

(2) 5. Sei $a = a_n a_{n-1}..a_2 a_1 a_0$ eine binärkodierte Zahl *mit* Vorzeichen und $a_n = 1$. Dann gilt

A. a ist negativ
B. a ist positiv
C. a ist gerade
D. a ist ungerade
E. a ist null
F. Keine der oben genannten Antworten

(2) 6. Sei $a = a_n a_{n-1}..a_2 a_1 a_0$ eine binärkodierte Zahl *ohne* Vorzeichen und $a_n = 1$. Dann gilt

A. a ist negativ
B. a ist positiv
C. a ist gerade
D. a ist ungerade
E. $a >= 2^{n-2}$
F. $a >= 2^n$
G. Keine der oben genannten Antworten

Module 7061: Grundlagen der Informatik
Final Exam I1p 1. Februar 2013 Name: _____

(1) 7. Sei $a = a_n a_{n-1}..a_2 a_1 a_0$ eine binärkodierte Zahl *mit* Vorzeichen und und $a_0 = 1$. Das bedeutet

A. a ist negativ
B. a ist positiv
C. a ist gerade
D. a ist ungerade
E. $a > 1$
F. $a < 1$
G. Keine der oben genannten Antworten

(1) 8. Die Bitweise Umkehrung aller Bits von $x = 10110$ ist das Zweierkomplement von x

A. Richtig
B. Falsch

(1) 9. Welcher Werte-Bereich wird mit 5 Bit im Zweierkomplement abgedeckt?

A. +15 bis -16
B. +16 bis -17
C. $+2^5$ bis -2^{5-1}
D. $+2^4 - 1$ bis $-2^4 + 1$
E. $+2^3 - 1$ bis $-2^3 - 1$
F. Keine der oben genannten Antworten

(5) 10. Bestimmen Sie Summe und Wert der Flags (Carry, Overflow, Sign, Zero).

(a)
```
  10010110
 +11110110
----------
```
8-Bit Summe:

Carry:..... Overflow:..... Sign:..... Zero: .....

(b)
```
  10100011
 +11001100
----------
```
8-Bit Summe:

Carry:..... Overflow:..... Sign:..... Zero: .....

(c)
```
  01010011
 +01101011
----------
```
8-Bit Summe:

Carry:..... Overflow:..... Sign:..... Zero: .....

(d)
```
  10110010
 +01001110
----------
```
8-Bit Summe:

Carry:..... Overflow:..... Sign:..... Zero: .....

(e)
```
  11111111
 +10101010
----------
```
8-Bit Summe:

Carry:..... Overflow:..... Sign:..... Zero: .....

Computer Architektur

(3) 11. Zeichne ein zu (a **AND** b) äquivalentes Schaltbild nur mit **NAND**s (Not AND)

Abbildung 1: Schaltbild eines **nand** Gatters

x	y	$\overline{x \wedge y}$	$\overline{x \wedge \overline{x \wedge y}}$	$\overline{\overline{x \wedge y} \wedge y}$
0	0	1	1	1
0	1	1	1	0
1	0	1	0	1
1	1	0	1	1

Abbildung 2: Zur freien Verfügung: Wahrheitstabellen verschiedener **nand** combinationen ($\wedge$ Symbol steht für das logische UND:

Abbildung 3: Skizze eines Circuits nur mit nand Gattern der $a \wedge b$ simuliert.

Module 7061: Grundlagen der Informatik
Final Exam I1p 1. Februar 2013 Name: _____

(3) 12. In Abbildung 4 sind zwei Speicherchips zu einem Schaltkreis verbunden. Das Signal `CS` (Chipselect, Chip-enable(=`CE`), dient für die (de-)aktivierung des Chips) können Sie ignorieren, da die beiden Speicherchips für die angegebene Sequenz immer aktiv sind. Es wird folgende Sequenz beobachtet:

Zeit	a4	a3	a2	a1	a0	d7	d6	d5	d4	d3	d2	d1	d0	rw	cs
1	1	0	0	1	0	1	0	1	0	1	1	1	1	1	0
2	0	0	0	0	1	1	0	1	0	0	1	1	1	1	0
3	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0

Beschreiben Sie zusammendfassend und in wenigen Worten (auch formal) was geschieht.

.....

.....

.....

.....

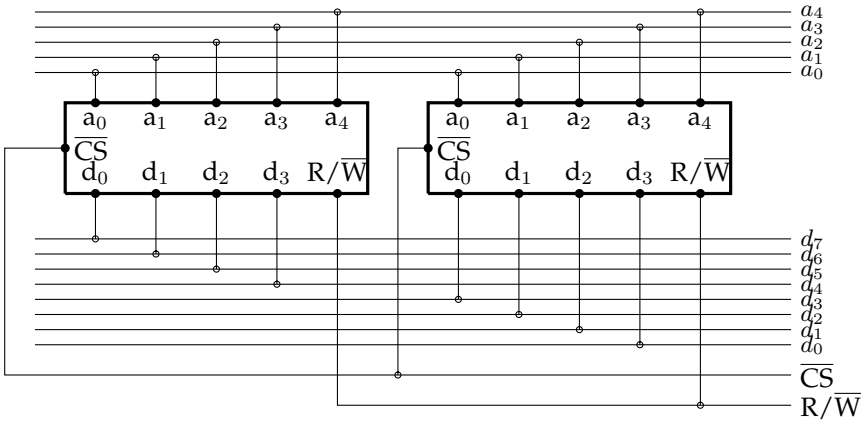

Abbildung 4: Circuit with two Memorychips

VonNeumann Architektur

(3) 13. Beschreiben Sie kurz und präzis was während eines Fetch-Execute-Zyklus geschieht. .....

(3) 14. Schreibe die μ-code Sequenz für die 6502 Instruktion `pha`. Die Fetch-Phase soll nicht aufgeschrieben werden. Der `PC` wurde bereits um eins hochgezählt und zeigt auf das Byte nach dem `pha` opcode.

(3) 15. Füllen Sie in Abbildung 5 die freien Plätze aus.

Adr.	Data	Data	Data	Labels	operator	operand
c000	a0	00			ldy	#$00
c002	a2	[ ]			ldx	#$00
c004	20	f0	ff		jsr	PLOT
c007	bc	29	c0		ldy	histogram,X
c00a				nextstar		
c00a	c0	00			cpy	[ ]
c00c	f0	[ ]			beq	nextline
c00e	a9	2a			lda	#"*"
[ ]	20	d2	ff		JSR	CHROUT
c013	88				dey	
c014	4c	0a	c0		jmp	nextstar
c017				nextline		
c017	a0	00			ldy	#$00
c019	e8				[ ]	
c01a	ec	55	c0		cpx	length
c01d	f0	09			beq	finish
c01f	[ ]	f0	ff		jsr	PLOT
c022	bc	29	c0		ldy	histogram,X
c025	4c	0a	c0		jmp	nextstar
c028				finish		
c028	60				rts	
⋮	⋮	⋮	⋮			

Abbildung 5: Assembler dump eines 6502 assembler programmes. Füllen Sie die sechs Lücken aus.

6502 Assembler

(1) 16. Was wird vom untenstehenden Program ausgedruckt?

```
    lda #2
    cmp #5
    bne no
    <print "ui">
    jmp yes
no:
    <print "hui">
    jmp last
yes:
    <print "pfui">
last:
    <print "lui">
```

Printout = ...

(1) 17. Sei Register X=`0x02`. Nach Ausführung der Instruktion `LDA $ABCD,Y` soll der Inhalt des Akkumulator A=`0xBE` sein. *Bestimme alle nötigen Speicherbelegungen* (Addresse und Inhalt) damit dies zutrifft. Benutze die Speichernotation **`MEM[<16Bit Adresse>] = <8Bit Inhalt>`**

. . . . . .

. . . . . .

. . . . . .

(1) 18. Sei Register X=`0x02`. Nach Ausführung der Instruktion `LDA $CD,X` soll der Inhalt des Akkumulator A=`0xBE` sein. *Bestimme alle nötigen Speicherbelegungen* (Addresse und Inhalt) damit dies zutrifft. Benutze die Speichernotation **`MEM[<16Bit Adresse>] = <8Bit Inhalt>`**

. . . . . .

. . . . . .

. . . . . .

(1) 19. Sei Register X=`0x02`. Nach Ausführung der Instruktion `LDA ($88,X)` soll der Inhalt des Akkumulator A=`0xBE` sein. *Bestimme alle nötigen Speicherbelegungen* (Addresse und Inhalt) damit dies zutrifft. Benutze die Speichernotation **`MEM[<16Bit Adresse>] = <8Bit Inhalt>`**

. . . . . .

. . . . . .

. . . . . .

(3) 20. Sei der Wert der Variable $a = 4$ und die Addresse von a `$ab12`. Sei der Wert der Variable $b = 7$ und die Addresse von b `$cdff`. Schreibe ein 6502/6510 Assembler Programm, das die Zuweisung $a = b + 1$ implementiert.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Stack

Im folgenden (Aufgaben 21-23) nehmen wir an das eine 16 Bit Address auf dem Stack wie folgt zu liegen kommt:

Addresse: `0xabcd`; Nach einem Push dieser Addresse resultiert folgender Stack (top element first): `0xcd`, `0xab`

(2) 21. Auf dem Stack liegen folgende Werte (top element first): `0x01`, `0x02`, `0x03`, `0x04`, `0x05`, `0x06`. An der Addresse `0xc000` steht die Instruktion `jsr 0xABCD`. Welchen Wert hat Register PC nach der Ausführung des Fetch-Exec-Cycles der Instruktion und was ist das Topelement?

PC = ... Topelement = ...

(2) 22. Auf dem Stack liegen folgende Werte (top element first): `0x01`, `0x02`, `0x03`, `0x04`, `0x05`, `0x06`. An der Addresse `0xc000` steht die Instruktion `rts`. Welchen Wert hat Register PC nach der Ausführung des Fetch-Exec-Cycles der Instruktion und was ist das Topelement?

PC = ... Topelement = ...

(2) 23. Bestimmen Sie den Werte von PC und Stack-Topelement nach folgender Programmsequenz:

```
LDA $AB
PHA
RTS
```

PC = ... Topelement = ...

Module 7061: Grundlagen der Informatik
Final Exam I1p 1. Februar 2013 Name: _____

C - The Language

(1) 24. Betrachte Listing 1. Welchen Wert wird die Funktion **afct** retournieren mit den Parameter `a=2, b=10`?

```
1
2  int afct(int a, int b){
3    int i, p;
4
5    p=1;
6
7    for(i=1;i<b;++i)
8      p = p *a;
9    return p;
10 }
```

Listing 1: loop over a product

`afct(2,10)` = ...

(3) 25. Gegeben sei `char hallo[10] = "Ahoi";`

Länge vom String in `hallo`: .....

`hallo[4]` = ...

`sizeof(hallo)` = ...

(1) 26. Schreiben Sie den Output des Programmes in Listing 2 auf: .....

```
1  void main()
2  {
3     printf("Hi!");
4     if (!(-5)){
5        printf("Bye");
6     }
7  }
```

Listing 2: Hello

(4) 27. Setzte die Klammern gemäss den Prioritäten die der C-Compiler setzt

(a) a = b == c

(b) a <= b | c * d < e

(c) c . e -> d . f == a . b

(d) * * b / * a * * c

(1) 28. Gegeben sind folgende C-Deklarationen. Kreise jeweils die entsprechenden Aussagen ein.

(a) `int *x(char);`

A. x ist eine Funktion mit einem Character-Pointer als Parameter, dass einen Pointer auf ein Integer zurückgibt
B. x ist ein Pointer auf ein Integer
C. x ist ein Pointer auf eine Funktion das einen Character-Pointer als Parameter hat und ein Integer zurückgibt
D. x ist ein Pointer auf eine Funktion mit einem Character als Parameter und einem Integer als Returnwert.
E. Keine der oben genannten Antworten

(b) `int (*x)(char);`

A. x ist ein Pointer auf eine Funktion das einen Character-Pointer als Parameter hat und ein Integer zurückgibt
B. x ist ein Pointer auf ein Character-Pointer
C. x ist eine Funktion mit einem Character-Pointer als Parameter, dass einen Pointer auf ein Integer zurückgibt
D. x ist ein Pointer auf eine Funktion mit einem Character als Parameter und einem Integer als Returnwert.
E. Keine der oben genannten Antworten

(c) `int *(*x[5]);`

A. x ist ein Pointer auf ein Array von 5 Integer-Pointer
B. x ist ein Array von 5 Pointer auf ein Integer
C. x ist ein Array von 5 Pointer auf Funktionen mit einem Integerparameter die ein Integer zurückgeben
D. x ist ein Pointer auf eine Funktion mit einem Arrayparameter von 5 Integer und einem Integer als Rückgabewert
E. Keine der oben genannten Antworten

(1) 29. Gegeben sind Aussagen über eine Variable x. Kreise jeweils die entsprechende Deklaration ein.

(a) x ist ein Pointer auf ein Integer

A. **int** *x;
B. **int*** x;
C. **int** (*x)();
D. **int** (*x);
E. Keine der oben genannten Antworten

(b) x ist ein Pointer auf eine Funktion die als Parameter einen Integer und einen Pointer auf ein Character hat und als Rückgabewert einen Integer hat

A. **int** *x(**int**, **char** *);
B. **int** *(*x)(**int**, **char** *);
C. **int** x(**int**, **char** *);
D. **int** (**x)(**int**, **char** *);
E. Keine der oben genannten Antworten

(c) x ist ein Pointer auf eine Funktion die als Parameter einen Integer hat und als Rückgabewert einen Pointer auf ein Integer hat

A. **int** (**x)(**int**);
B. **int** *x(**int**);
C. **int** **x(**int**);
D. **int** (*x)(**int**);
E. Keine der oben genannten Antworten

C - Programmieren

(1) 30. Schreiben Sie den Output des Programmes in Listing 3 auf: .....

```
1  void main()
2  {
3    int a=1,b=2,c=3,d=4;
4    if (d > c)
5       if (c < b)
6          printf("%d␣%d",d,c);
7       else if (c > a)
8          printf("%d␣%d",c,d);
9    if (c > a)
10         if (b < a)
11            printf("%d␣%d",c,a);
12         else if (b < c)
13            printf("%d␣%d",b,c);
14 }
```

Listing 3: if/then/else

(1) 31. Welchen Output erzeugt das Programm in Listing 4?

Output of Listing 4 = ...

```
1  void main()
2  {
3    int i=4;
4    for (i=3; i>1; i--){
5      printf("%d␣",i);
6    }
7    printf("%d␣",i);
8  }
```

Listing 4: for loop

(1) 32. Schreiben Sie den Output des Programmes in Listing 5 auf: .....

```
1  int main()
2  {
3    int i,j;
4    for (i=2; i<5; i++){
5      if(i==3)
6        continue;
7
8      for (j=3; j<5; j++){
9          if(i==j)
10             break;
11
12         printf("%d␣%d␣", i, j);
13       }
14
15     printf("%d␣%d␣", i, j);
16   }
17   return 0;
18 }
```

Listing 5: Break or continue

(1) 33. Welchen Output erzeugt das Programm in Listing 6?

```
1  void main()
2  {
3    int val1=11;
4    int val2=22;
5    int* ptr1;
6    int* ptr2;
7    ptr1 = &val1;
8    ptr2 = ptr1;
9    *ptr1 += 1;
10   *ptr2 = *ptr2 + 1;
11
12   printf("val1 = %d \n val2 = %d \n *ptr1 = %d \n *ptr2 = %d\n", val1,
        val2, *ptr1, *ptr2); }
```

Listing 6: Pointing to Values

Output of Listing 6:

val2 = ... val1 = ...

*ptr1 = ... *ptr2 = ...

(1) 34. Schreiben Sie den Output des Programmes in Listing 7 auf: .....

```
1  void main()
2  {
3    int val=2;
4    int* ptr;
5    ptr = &val;
6    val = 3;
7    *ptr = *ptr + 2;
8    val = val + 3;
9    printf("%d %d", val, *ptr);
10 }
```

Listing 7: More pointers

(2) 35. Schreiben Sie den Output des Programmes in Listing 8 auf: .....

```
int a,b;
int *ap, *bp;
int ** cp, **dp;
int *ep, *fp;

void xy(int ***p, int ***q){
  int ***tmp;
  tmp = **p;
  **p = **q;
  **q = tmp;
}

void main()
{
  a = 2;
  b = 3;
  ap = &a;
  bp = &b;
  cp = &ap;
  dp = &bp;
  ep = ap;
  fp = bp;
  xy(&cp, &dp);

  printf("%d␣%d␣", a, b);
  printf("%d␣%d␣", *ap, *bp);
  printf("%d␣%d␣", **cp, **dp);
  printf("%d␣%d␣", *ep, *fp);
}
```

Listing 8: Pointers to pointer

(2) 36. Auszug aus dem ISO/IEC 9899 C Standart Dokument:

> " The grouping of an expression does not completely determine its evaluation. In the following fragment
>
> ```
> #include <stdio.h>
> int sum;
> char *p;
> /* ... */
> sum = sum * 10 - '0' + (*p++ = getchar());
> ```
>
> the expression statement is grouped as if it were written as
>
> ```
> sum = (((sum * 10) - '0') + ((*(p++)) = (getchar())));
> ```
>
> but the actual increment of p can occur at any time between the previous sequence point and the next sequence point (the ;), and the call to getchar can occur at any point prior to the need of its returned value."

Welche sind die möglichen Ausgaben des C-Programms in Listing 9?

```
1 #include <stdio.h>
2 int a=0,b=0;
3 int z=0;
4 (a = z++) || (b= z++) ;
5 printf("a=%d␣b=%d␣\n", a, b);
```

Listing 9: Order of post-increment evaluation

Möglicher Output:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

(3) 37. Kopie einer Frage aus einem C-Programmier-Forum:

This is my code to add node at beginning.

```
1  void screate(ll *node)
2  {
3    ll *newNode=(ll *)malloc(sizeof(ll));
4    printf("Enter␣number␣:\t");
5    scanf("%d",&newNode->data);
6    if(newNode->data != NULL)
7    {
8      newNode->next=node;
9      node= newNode;
10     screate(node);
11   }
12   else
13   {
14     free(newNode);
15     newNode=NULL;
16   }
17 }
```

Listing 10: Code snippet to add node at the beginning of a list

This is the current node

```
56->78->77->NULL
```

But, when i'm trying to add new node at the beginning, then still I'm getting same output i.e. 56->78->77->NULL. Need Help !!

Geben Sie dem C-Anfänger eine Antwort (kurz und präzis) auf sein Problem:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .