

Question 1:

a) Hex: 0xDEADBEEF Given

Binary = 1101 1110 1010 1101 1011 1110 1110 1111

Decimal:  $(15 \times 16^0) + (14 \times 16^1) + (10 \times 16^2) + (11 \times 16^3) + (3 \times 16^4) + (10 \times 16^5)$   
 (from Hex)  $+ (14 \times 16^6) + (13 \times 16^7)$

$$= 15 + 224 + 3584 + 45056 + 8,51,968 + 1,04,85,760 + 23,48,81,024 + 3,48,96,60,928$$

$$= 3,735,928,559$$

Octal from Binary = 011 011 110 101 011 011 011 111 011 101 111

⇒ 3 3 6 5 3 3 7 3 5 7

Data type = uint32\_t

b) Given: Decimal: 310422

2	310,422	0
2	155,211	1
2	77,605	1
2	38,802	0
2	19,401	1
2	9,700	0
2	4,850	0
2	2,425	1
2	1,212	0
2	606	0
2	303	1
2	151	1
2	75	1
2	37	1
2	18	0
2	9	1
2	4	0
2	2	0
	1	1

Binary: 0100101110010010110

Hex: 4 8 C 9 6 i.e 0x00048C96

Octal: 001 001 011 110 010 010 110

⇒ 1 1 3 6 2 2 6

Data type: uint32\_t, int32\_t

c) 1465 - Decimal

Binary:

2	1465	1
2	732	0
2	366	0
2	183	1
2	91	1
2	45	1
2	22	0
2	11	1
2	5	1
2	2	0
	1	1

Binary = 01011011001

Hex = 5B9 i.e. 0x5B9

Octal: 010 110 11001  
: 2 6 71

Data type: uint16-t, int16-t

d) Given: 0b 111010101110

Decimal:  $(0 \times 1) + (1 \times 2) + (1 \times 4) + (1 \times 8) + (0 \times 2^7) + (1 \times 2^5) + (0 \times 2^6)$   
 $+ (1 \times 2^7) + (0 \times 2^8) + (1 \times 2^9) + (1 \times 2^{10}) + (1 \times 2^{11})$   
 $= 0 + 2 + 4 + 8 + 0 + 32 + 0 + 128 + 0 + 512 + 1024 + 2048$   
 $= 3758$

Hex: 1110 1010 1110  
 $\Rightarrow$  0xEAE i.e. 0x0EAE

Octal: 111 010 101 110  
 $\Rightarrow$  7256

Data type: uint16-t, int16-t

e) Given: 0x8FA44

decimal:  $(4 \times 16^0) + (4 \times 16^1) + (10 \times 16^2) + (15 \times 16^3) + (8 \times 16^4)$   
 $= 588356$

Binary  $\rightarrow$  1000 1111 1010 0100 0100  
 (from hex)

Octal  $\rightarrow$  010 001 111 101 001 000 100  
 $\Rightarrow$  2175104

Data type: uint32-t, int32-t

f) Given decimal: -172  
data type - int 16-t

Binary: 
$$\begin{array}{r|l} 2 & 172 & 0 \\ \hline 2 & 86 & 0 \\ \hline 2 & 43 & 1 \\ \hline 2 & 21 & 1 \\ \hline 2 & 10 & 0 \\ \hline 2 & 5 & 1 \\ \hline 2 & 2 & 0 \\ \hline & 1 & 1 \end{array}$$

$\Rightarrow$  0000 0000 1010 1100  
1<sup>st</sup> complement: 1111 1111 0101 0011  
2<sup>nd</sup> complement: 1111 1111 0101 0100

Hex: 0xFF54

Octal: 001 111 111 101 010 100  
 $\Rightarrow$  177524

g) Given decimal: 172  
data type: uint 16-t, int 16-t, uint 8-t

Binary: 
$$\begin{array}{r|l} 2 & 172 & 0 \\ \hline 2 & 86 & 0 \\ \hline 2 & 43 & 1 \\ \hline 2 & 21 & 1 \\ \hline 2 & 10 & 0 \\ \hline 2 & 5 & 0 \\ \hline 2 & 2 & 0 \\ \hline & 1 & 1 \end{array}$$

$\Rightarrow$  0000 1010 1100 1100

Hex: 0x0AC

Octal: 010 101 100  
 $\Rightarrow$  954

Table:

Decimal	Binary	Hexadecimal	Hexadecimal	data type
8,135,928,559	0b110111101010110110111101110111	0xDEADBEEF		uint 32-t
310,422	0b01001011110010010110	0x48C96		uint 32-t, int 32-t
1765	0b010110111001	0x5B9		uint 16-t, int 16-t
3758	0b111010101110	0xEAE		uint 16-t, int 16-t
588,356	0b1000111101001000100	0x8FA44		uint 32-t, int 32-t
-172	0b111111101010100	0xFF54		int 16-t
172	0b10101100	0xAC		uint 16-t, int 16-t, uint 8-t

## Question 2:

• 124.84375

$$\begin{array}{r} 2 \overline{) 124} \quad 0 \\ 2 \overline{) 62} \quad 0 \\ 2 \overline{) 31} \quad 1 \\ 2 \overline{) 15} \quad 1 \\ 2 \overline{) 7} \quad 1 \\ 2 \overline{) 3} \quad 1 \\ 1 \end{array}$$

0111100

$$\begin{array}{r} 0.84375 \\ \times 2 \\ \hline 1.68750 \quad 1 \\ 0.68750 \\ \times 2 \\ \hline 1.37500 \quad 1 \\ 0.37500 \\ \times 2 \\ \hline 0.75000 \quad 0 \\ 0.75 \\ \times 2 \\ \hline 1.50 \quad 1 \\ 0.50 \\ \times 2 \\ \hline 1.0 \quad 1 \end{array}$$

⇒ 11011

fixed point: 0111100.11011000

32 bit precision:

1.11110011011000 × 2<sup>6</sup>

exp: 6 + 127 = 133

signbit exponent mantissa

⇒ 10000101

0 - 10000101 - 111100110110000000000000

$$\begin{array}{r} 2 \overline{) 133} \quad 1 \\ 2 \overline{) 66} \quad 0 \\ 2 \overline{) 33} \quad 1 \\ 2 \overline{) 16} \quad 0 \\ 2 \overline{) 8} \quad 0 \\ 2 \overline{) 4} \quad 0 \\ 2 \overline{) 2} \quad 0 \\ 1 \end{array}$$

• -12.45897

$$\begin{array}{r} 2 \overline{) 12} \quad 0 \\ 2 \overline{) 6} \quad 0 \\ 2 \overline{) 3} \quad 1 \\ 1 \end{array}$$

$$\begin{array}{r} 0.45897 \\ \times 2 \\ \hline 0.91794 \quad 0 \\ 1.83588 \quad 1 \\ 0.83588 \\ \times 2 \\ \hline 1.67176 \quad 1 \\ 0.67176 \\ \times 2 \\ \hline 1.34352 \quad 1 \\ 0.34352 \\ \times 2 \\ \hline 0.68704 \quad 0 \\ 1.37408 \quad 1 \\ 0.37408 \\ \times 2 \\ \hline 0.74816 \quad 0 \\ 1.49632 \quad 1 \end{array}$$

⇒ 00001100

2's complement ⇒

$$\begin{array}{r} \cancel{1110000} \\ 1111101 \\ \Rightarrow 11110011 \\ 11110100 \end{array}$$

fixed point:

⇒ 11110100.01110101

⇒ 01110101

32 bit precision: 1.11101000110101 × 2<sup>7</sup>

→ To be continued

## Question 2

•  $-12.45897$

32 bit floating point

12 in binary 0000 1100

0.45897 binary 0111 0101

$12.45897 = 1100.01110101$

$= 1.1000110101 \times 2^3$

sign bit exp Mantissa

1 - 1 000 0010 - 1000 1110 1011 1111 1111 1111

exp = 3 + 127  
= 130

$$\begin{array}{r} 2 \overline{) 130} \quad 0 \\ \underline{2 \quad 60} \quad 1 \\ 2 \overline{) 60} \quad 0 \\ \underline{2 \quad 30} \quad 0 \\ 2 \overline{) 30} \quad 0 \\ \underline{2 \quad 15} \quad 0 \\ 2 \overline{) 15} \quad 0 \\ \underline{2 \quad 7} \quad 0 \\ 2 \overline{) 7} \quad 0 \\ \underline{2 \quad 3} \quad 0 \\ 2 \overline{) 3} \quad 0 \\ \underline{2 \quad 1} \quad 0 \\ 2 \overline{) 1} \quad 0 \\ \underline{2 \quad 0} \quad 0 \end{array}$$

## Question 3

• 0xC41-FECO

1010

1100 0100 0100 0001 1111 1110 1100 0000

sign bit = 1

Mantissa: 1.100 0001 1111 1101 1100

exponent: 1000 1000  $\Rightarrow 2^7 + 2^3$   
 $\Rightarrow 128 + 8 = 136$   
 $= 136 - 127 = 9$

11000.0011111101100

$\Rightarrow -24.$

$\Rightarrow 1100000111.1111101100$

$2^9 + 2^8 + 4 + 2 + 1$   
 $= 512 + 256 + 4 + 2 + 1$

$= 775$

fraction:  $(1 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3})$   
 $+ (1 \times 2^{-4}) + (1 \times 2^{-5}) + (0 \times 2^{-6})$   
 $+ (1 \times 2^{-7}) + (1 \times 2^{-8}) + (0 \times 2^{-9}) + (0 \times 2^{-10})$

$\Rightarrow 0.5 + 0.25 + 0.125 + 0.0625 + 0.0313 + 0.0078 + 0.0039$

$\Rightarrow 0.9805$

$\Rightarrow -775.9805$

$=$

X wrong calculation

$\Rightarrow \text{integral} = (1 \times 2^4) + (1 \times 2^3) = 24$

fraction:  $(0 \times 2^{-1}) + (0 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4})$   
 $+ (1 \times 2^{-5}) + (1 \times 2^{-6}) + (1 \times 2^{-7}) + (1 \times 2^{-8})$   
 $+ (1 \times 2^{-9}) + (1 \times 2^{-10}) + (0 \times 2^{-11}) + (1 \times 2^{-12})$   
 $+ (1 \times 2^{-13})$

$= 0.1250 + 0.0625 + 0.0313 + 0.0156 +$   
 $+ 0.0078 + 0.0039 + 0.0020 + 0.0010 +$   
 $0.0002 + 0.0001$   
 $= 0.2494$

### Question 3

• 0x46FE - 8000

0100	0110	1111	1110	1000	0000	0000	0000
sign bit	exponent		Mantissa				
0	10001101		11111101000000000000				

$$\text{exponent} = 2^7 + 2^3 + 2^2 + 2^0$$

$$\Rightarrow 128 + 8 + 4 + 1$$

$$= 141$$

$$\Rightarrow 141 - 127 = 14$$

$$L. 111111010000000000000000 \times 2^{14}$$

$$\Rightarrow 11111101000000.00000000$$

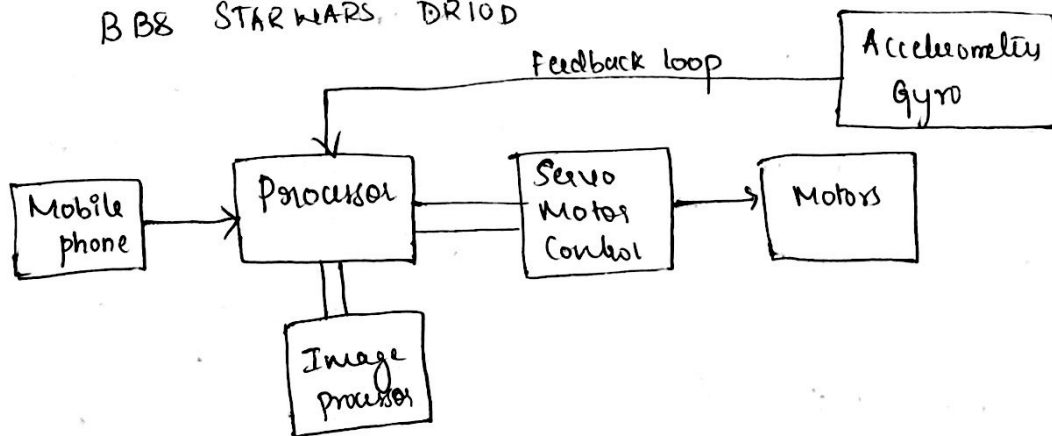
$$\Rightarrow 2^{14} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^6$$

$$= 16384 + 8192 + 4096 + 2048 + 1024 + 512 + 256 + 64$$

$$= 32576.0$$

### Question 4:

BBS STAR WARS DROID



# Question 5:

At 100, Data = 0xFE

After

0xF1 & 127

$$\begin{array}{r} \Rightarrow 11110001 \\ 01111111 \\ \hline 01110001 \end{array}$$

$\Rightarrow 0x71$

127 to Binary

$$\begin{array}{r} \Rightarrow 2 \overline{)127} \quad | \\ 2 \overline{)63} \quad | \\ 2 \overline{)31} \quad | \\ 2 \overline{)15} \quad | \\ 2 \overline{)7} \quad | \\ 2 \overline{)3} \quad | \\ 1 \end{array}$$

$\Rightarrow 01111111$

At 101  $\Rightarrow$  Data = 0x34

0x34 = 00110100 in Binary

0x34 + 17 (Decimal)

17  $\Rightarrow$  00010001 in Binary

So,

$$\begin{array}{r} 00110100 \\ 00010001 \\ \hline 01000101 \end{array}$$

$\Rightarrow 0x45$

At 103  $\Rightarrow$  15%4

$$\Rightarrow 4 \overline{)15} \\ 3$$

Remainder = 3

$\therefore \Rightarrow 0x03$

At 102  $\Rightarrow$  Data = 0x8C

$$\begin{array}{r} \Rightarrow \cancel{10001101} \gg 4 \quad 01001101 \gg 4 \\ \Rightarrow \cancel{00000000} \quad 00000100 \end{array}$$

$\Rightarrow 0x08$

At 104  $\Rightarrow$  Data = 0x61

After  $\Rightarrow 0x61$

At 105  $\Rightarrow$  Data = 0x28

After : 12251422

$$1225 \Rightarrow 00000001$$

So, 00100000

$$4222 \Rightarrow 00000100$$

So, 00010000

At 105,

$$\begin{array}{r} 0010\ 0000 \\ 0001\ 0000 \\ \hline 0011\ 0000 \\ \hline \quad 3 \quad \quad 0 \end{array}$$

$\Rightarrow$  0x30

At 106  $\rightarrow$  Data = 0x23

After = 0x23

At 107  $\rightarrow$  Data = 0x40

After = 22

in binary  $\rightarrow$   $\frac{0001\ 0110}{1\quad 6}$

$\Rightarrow$  0x16

Address	Data	Data After
100	0xFE	0x71
101	0x34	0x45
102	0x8C	0x08
103	0x40	0x03
104	0x61	0x61
105	0x28	0x30
106	0x23	0x23
107	0x40	0x16



Question 6:

```
#include<stdio.h>

int main()
{
    unsigned char arr [8];
    int index=0;
    arr[0] = 0xFE;
    unsigned char * ptr = arr;
    printf("Addr: %d\t%x\t", index, arr[index]);

    arr[0]= 0xF1 & 127;

    printf("%x\n", arr[index]);

    index++;

    arr[1]=0x34;
    printf("Addr: %d\t%x\t", index, arr[index]);

    arr[1]+=17;
    printf("%x\n", arr[index]);

    index+=2;

    arr[3]=0x61;
    printf("Addr: %d\t%x\t", index, arr[index]);

    arr[3]=15%4;
    printf("%x\n", arr[index]);

    index--;
    arr[2]=0x8C;
    printf("Addr: %d\t%x\t", index, arr[index]);
    arr[2]>>=4;
    printf("%x\n", arr[index]);

    index=4;
    arr[4]=0x61;
    printf("Addr: %d\t%x\t", index, arr[index]);
    printf("%x\n", arr[index]);

    index=5;
    arr[5]=0x28;
    printf("Addr: %d\t%x\t", index, arr[index]);
```

```

arr[5]= (1<<5)|(4<<2);
printf("%x\n", arr[index]);

index=6;
arr[6]=0x23;
printf("Addr: %d\t%x\t", index, arr[index]);
printf("%x\n", arr[index]);

index=7;
arr[7]=0x40;
printf("Addr: %d\t%x\t", index, arr[index]);
arr[7]=22;
printf("%x\n", arr[index]);

return 0;
}

```

Output:

Addr: 0	fe	71
Addr: 1	34	45
Addr: 3	61	3
Addr: 2	8c	8
Addr: 4	61	61
Addr: 5	28	30
Addr: 6	23	23
Addr: 7	40	16

## Question 7:

### OUTPUT:

```
theja@Serenity:~  
Permission denied (publickey).  
theja@Serenity:~$ ssh root@192.168.7.2  
^C  
theja@Serenity:~$ ssh root@192.168.7.2  
The authenticity of host '192.168.7.2 (192.168.7.2)' can't be established.  
ECDSA key fingerprint is 74:64:4f:6f:e5:9b:5f:40:c4:68:24:bd:2f:5a:27:6d.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.7.2' (ECDSA) to the list of known hosts.  
Debian GNU/Linux 7  
BeagleBoard.org Debian Image 2015-03-01  
Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack\_Debian  
default username:password is [debian:tempwd]  
root@beaglebone:~# vim main.c  
root@beaglebone:~# gcc main.c -o main  
main.c:27:1: error: expected ';' before '}' token  
root@beaglebone:~# vim main.c  
root@beaglebone:~# gcc main.c -o main  
root@beaglebone:~# ./main  
The size of char:1The size of int:4The size of float:4The size of double:8The size of short:2The size of long:4The size of long int:4The size of long long:8The size of in  
t8_t:1The size of uint8_t:1The size of uint16_t:2The size of uint32_t:4The size of char*:4The size of int*:4The size of float*:4The size of void*:4The size of int8_t*:4Th  
e size of int16_t*:4The size of int32_t*:4root@beaglebone:~# vim main.c  
root@beaglebone:~# gcc main.c -o main  
root@beaglebone:~# ./main  
The size of char:1  
The size of int:4  
The size of float:4  
The size of double:8  
The size of short:2  
The size of long:4  
The size of long int:4  
The size of long long:8  
The size of int8_t:1  
The size of uint8_t:1  
The size of uint16_t:2  
The size of uint32_t:4  
The size of char*:4  
The size of int*:4  
The size of float*:4  
The size of void*:4  
The size of int8_t*:4  
The size of int16_t*:4  
The size of int32_t*:4  
root@beaglebone:~#
```

### CODE:

```
theja@Serenity:~  
#include <stdio.h>  
#include <stdint.h>  
  
int main()  
{  
    printf("The size of char:%d\n",sizeof(char));  
    printf("The size of int:%d\n",sizeof(int));  
    printf("The size of float:%d\n",sizeof(float));  
    printf("The size of double:%d\n",sizeof(double));  
    printf("The size of short:%d\n",sizeof(short));  
    printf("The size of long:%d\n",sizeof(long));  
    printf("The size of long int:%d\n",sizeof(long int));  
    printf("The size of long long:%d\n",sizeof(long long));  
    printf("The size of int8_t:%d\n",sizeof(int8_t));  
    printf("The size of uint8_t:%d\n",sizeof(uint8_t));  
    printf("The size of uint16_t:%d\n",sizeof(uint16_t));  
    printf("The size of uint32_t:%d\n",sizeof(uint32_t));  
    printf("The size of char*:%d\n",sizeof(char*));  
    printf("The size of int*:%d\n",sizeof(int*));  
    printf("The size of float*:%d\n",sizeof(float*));  
    printf("The size of void*:%d\n",sizeof(void*));  
    printf("The size of int8_t*:%d\n",sizeof(int8_t*));  
    printf("The size of int16_t*:%d\n",sizeof(int16_t*));  
    printf("The size of int32_t*:%d\n",sizeof(int32_t*));  
  
    return 0;  
}
```

"main.c" 27L, 1020C

1,1

#### Question 8: Pseudo code

If the length is negative

Print "enter a positive integer"

If the length is a character

Print "enter a valid integer"

If the length is less than or greater than the string length

Print "enter the appropriate length of the string"

If the string contains only the null character

Print "enter a valid string"

If the string contains only white spaces

Print "enter a valid string"

If the string contains letters, digits , special characters or combination of all of them

Print "reverse operation successful "

#### Question 9:

```
#include <stdio.h>
char reverse(char *str, int len);
int main(void)
{
    char word[100];
    int length = 0;
    printf("Enter a string: ");
    //scanf("%s",word);
    gets(word);
    while(word[length]!='\0')
    {
        length++;
    }

    reverse(word,length);
}

char reverse(char *str, int len)
{
    int count; int i=0;
    char *begin, *last, *k, temp, temp2, *space;
    int l=0;

    if(len==0)
    {
```

```

        printf("Please enter a string.\n");
        return 1;
    }

    if(*str=="\0")
    {
        printf("Please enter a valid string");
        return 1;
    }

    space=str;
    while(i<len)
    {
        if(*space == ' ')
        {
            count++;
        }
        i++;
        space++;
    }

    if(count == len)
    {
        printf("Please enter a valid string");
        return 1;
    }

    begin=str;
    last=str+len-1;
    k=str;

    while(begin<last)
    {
        temp=*begin;
        *begin=*last;
        *last=temp;
        last--;
        begin++;
        l++;
    }

    printf("Reverse is: ");
    while(*k!='\0')
    {
        printf("%c", *k);
        k++;
    }

    return 0;
}

```