Computer Science 250

Homework 1

→ Question 01:

(A) $+47_{10}$ to 8-bit 2s compl. int. in binary and hexadecimal.

• To find $+47_2$, I am using the base 2 to do the conversion:

$47 - \underline{32} = 15 - \underline{8} = 7 - \underline{4} = 3 - \underline{2} = 1 - \underline{1} = 0.$

Then, $32+8+4+2+1 = 47$. Therefore,
$2^5 \cdot 1 + 2^3 \cdot 1 + 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 1 = 47.$

2^5	2^4	2^3	2^2	2^1	2^0
↓	↓	↓	↓	↓	↓
1	0	1	1	1	1

$\therefore +47_{10} = 101111_2$; in 8-bit we get $\boxed{00101111}$.
Since the # is positive, no further action is needed in 2s compl.

• Finding $+47_{10}$ in hexadecimal:

0 1 2 3 4 5 6 7 8 9 A B C D E F
(10 11 12 13 14 15)

	Remainder	
$47 \div 16 = 2 +$	15	→ f
$2 \div 16 = 0 +$	2	→ 2

Then, $+47_{10} = 0x002F$.

(B) -13_{10}.

• -13_{10} to 8-bit 2s ci:

$13 - \underline{8} = 5 - \underline{4} = 1 - \underline{1} = 0$

Then, $8+4+1 = 13$;
$2^3 + 2^2 + 2^0 = 13.$

2^3	2^2	2^0
↓	↓	↓
1	1	1

→ $|-13_{10}|$ into 13_2 = 00001101 (8-bit).

• Inverting to find 2's compl. to -13_{10}:

$11110010 + 1 = \boxed{11110011}$.

• -13_{10} to hexadecimal:

$2^3\ 2^2\ 2^1\ 2^0$
1 1 1 1 0 0 1 1

15 3
f 3

Then, $0x00F3$.

(C) $+47.0_{10}$ to 32-bit IEEE.

0 | 10000100 | 01111 0000000000000000000

Sign (+ number) | 8 exp. bit

↳ $+47.0_{10}$ to binary 8-bit from part A is: 00101111.

$47.0 = 4.7 \cdot 10^1$

$00101111 = 1.01111 \cdot 2^5$

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

128 64 32 16 8 4 2 1

$127 + 5 = 132$

$132 - \frac{128}{2^7} = 4 - \frac{4}{2^2} = 0.$

$132_{10} = 10000100_2$

Finally, $+47.0_{10}$ to 32-bit IEEE is

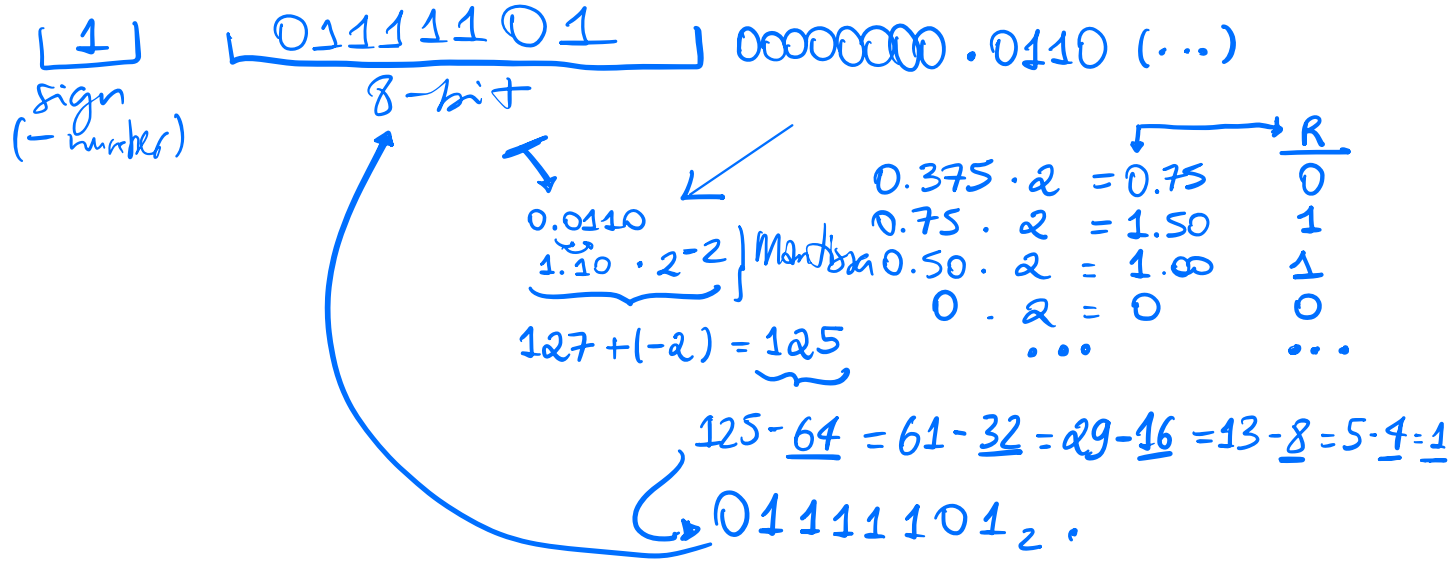
0100 0010 0011 1100 0000 0000 0000 0000

4 2 3 12 0 0 0 0

C

$+47.0$ in hex. is $0x423C0000$.

(D) -0.375_{10} to 32-bit IEEE.



Finally, -0.375_{10} to 32-bit IEEE is

1011 1110 1100 0000 0000 0000 0000 0000

11 14 12 0 0 0 0 0

B E C

Then, -0.375_{10} to hex. is 0xBEC00000.

(E) String for 250!

ASCII hex:

53 74 72 69 6E 67 20 66 6F 72 20 31 35 30 21.

(F) The 32-bit signed says that it ranges from $-2^{31}-1$ to $+2^{31}-1$. Then, for example, -2^{32} and 2^{32} would not be compatible with this rule; then $\pm 4294967296_{10}$ cannot be represented as they are out of range.

→ Question 02:

A)

(A) a lives in the stack (local);

(B) b_ptr lives in the stack (pointer);

(C) *b_ptr lives in the heap;

(D) e_ptr lives in global data (global var);

(E) *e_ptr lives in the stack.

B)

```
float* e_ptr;

float foo(float* x, float *y, float* z){
    if (*x > *y + *z){
        return *x;
    } else {
        return *y+*z;
    }
}

int main() {
    float a = 1.2;
    e_ptr = &a;
    float* b_ptr = (float*) malloc (2*sizeof(float));
    b_ptr[0] = 7.0;
    b_ptr[1] = 4.0;
    float c = foo(e_ptr, b_ptr, b_ptr+1);
    free(b_ptr);
    if (c > 10.5){
        return 0;
    } else {
        return 1;
    }
}
```

1.2 > 7.0 + 5

1.2 > 12

7+5

12

1.2, 7.0, 4.0+1

12 > 10.5

return 0

The value returned by main is 0.

→ Question 03:

```
adeildovieira@MacBook-Air-de-Adeildo homework-1-c % time ./myProgramUnopt
C[111][392]=-1801792042
./myProgramUnopt  0.27s user 0.00s system 98% cpu 0.280 total
adeildovieira@MacBook-Air-de-Adeildo homework-1-c % time ./myProgramOpt
C[111][392]=-1801792042
./myProgramOpt  0.12s user 0.01s system 38% cpu 0.333 total
adeildovieira@MacBook-Air-de-Adeildo homework-1-c %
```

for the unopt I got a rt of 0.27s, meanwhile the optimized ran for 0.12s.

The optimized version is $\frac{0,27-0,12}{0,27}$. 100% ⇔ 55,56% faster.