→ int main( )

int num[5];

"hello";

}

character array + null character

" " "%d \n"

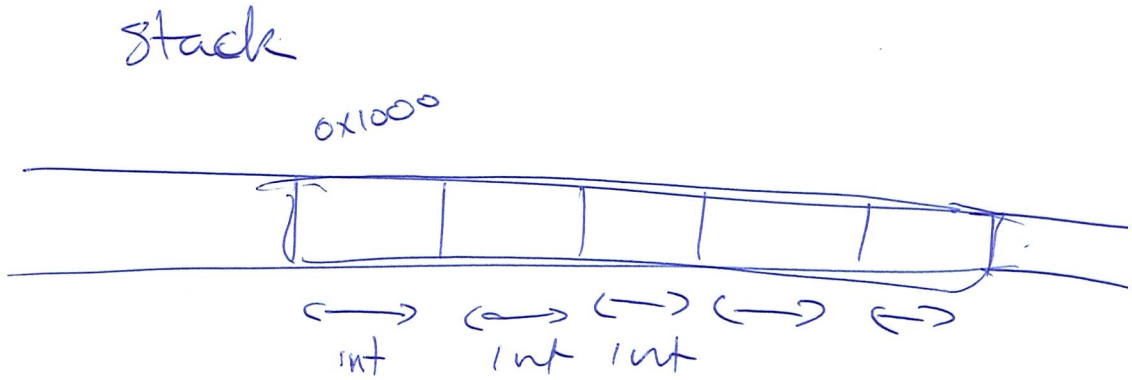
---

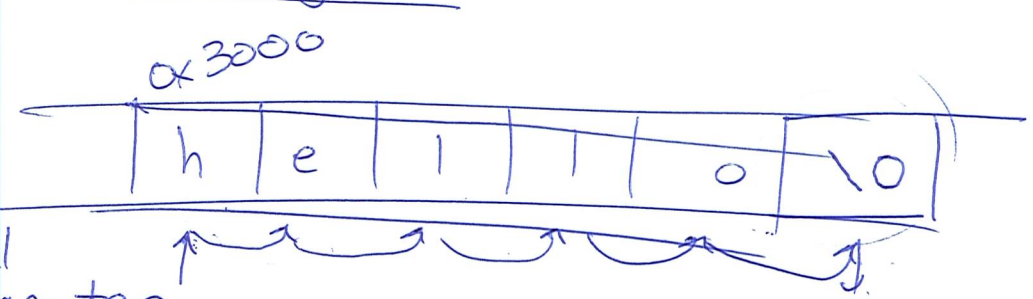
printf("%s\n", "hello");

{

char rowing[ ] = "rowing";

}

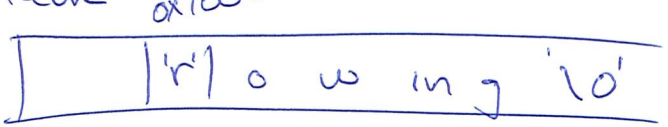




0x3000

"hello"

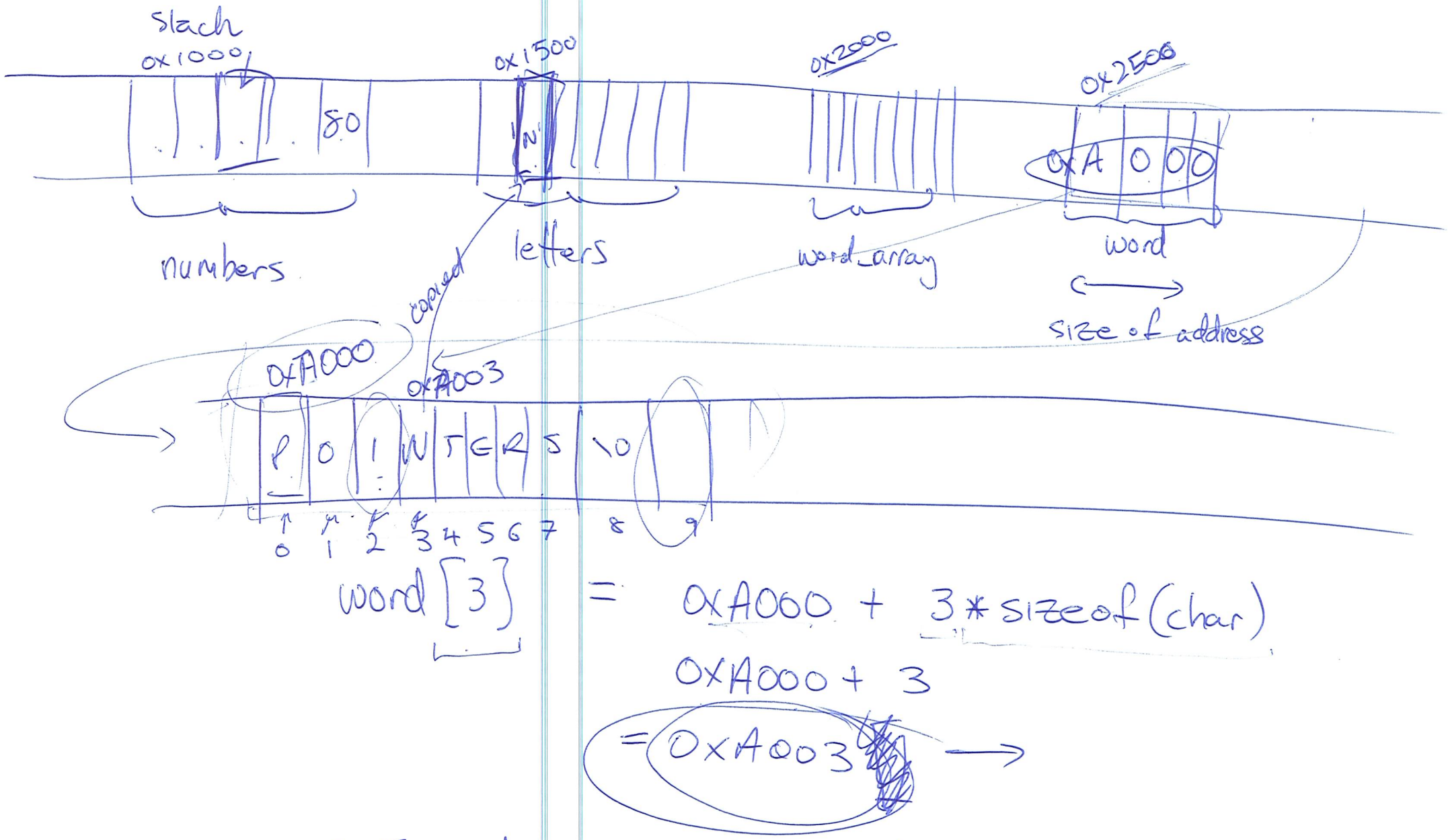
char [] name = {'M', 'A', 'R', 'Y'};



7 bytes

char [] x = name;

```c
#include <stdio.h>
#include <string.h>

int main() {

    int numbers[5];

    char letters[7];

    char word_array[] = { 'P','O','I','N','T','E','R','S' };

    char *word = "POINTERS";

    letters[1] = word[3];

    numbers[4] = word[0];

    printf("numbers[2] = %d\n", numbers[2]);
    printf("numbers[4] = %d\n", numbers[4]);
    printf("letters[1] = %d\n", letters[1]);
    printf("word_array[2] = %d\n", word_array[2]);
    printf("word[9] = %d\n", word[9]);

    int i;
    for (i = 0; i < strlen(word); ++i)
        printf("%c = %d\n", word[i], word[i]);

    /*
     P = 80
     O = 79
     I = 73
     N = 78
     T = 84
     E = 69
     R = 82
     S = 83
     */

    return 0;
}
```



word[3] = 0xA000 + 3 * sizeof(char)

0xA000 + 3

= 0xA003 →

0x1501 value = 0xA003 value

char msg[] = "hello!";

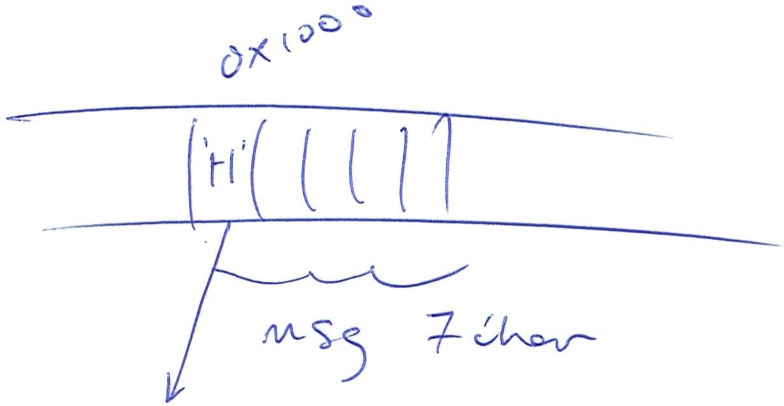


Figure A

char * str = &msg[0];

*(0x1000) → 'H' (73)

*(0x1001) → 'E'

*(0x1000)+1 → 74 'I'
73

*(0x1000+1) → 'E'

Please note my mistake

boolean expressions

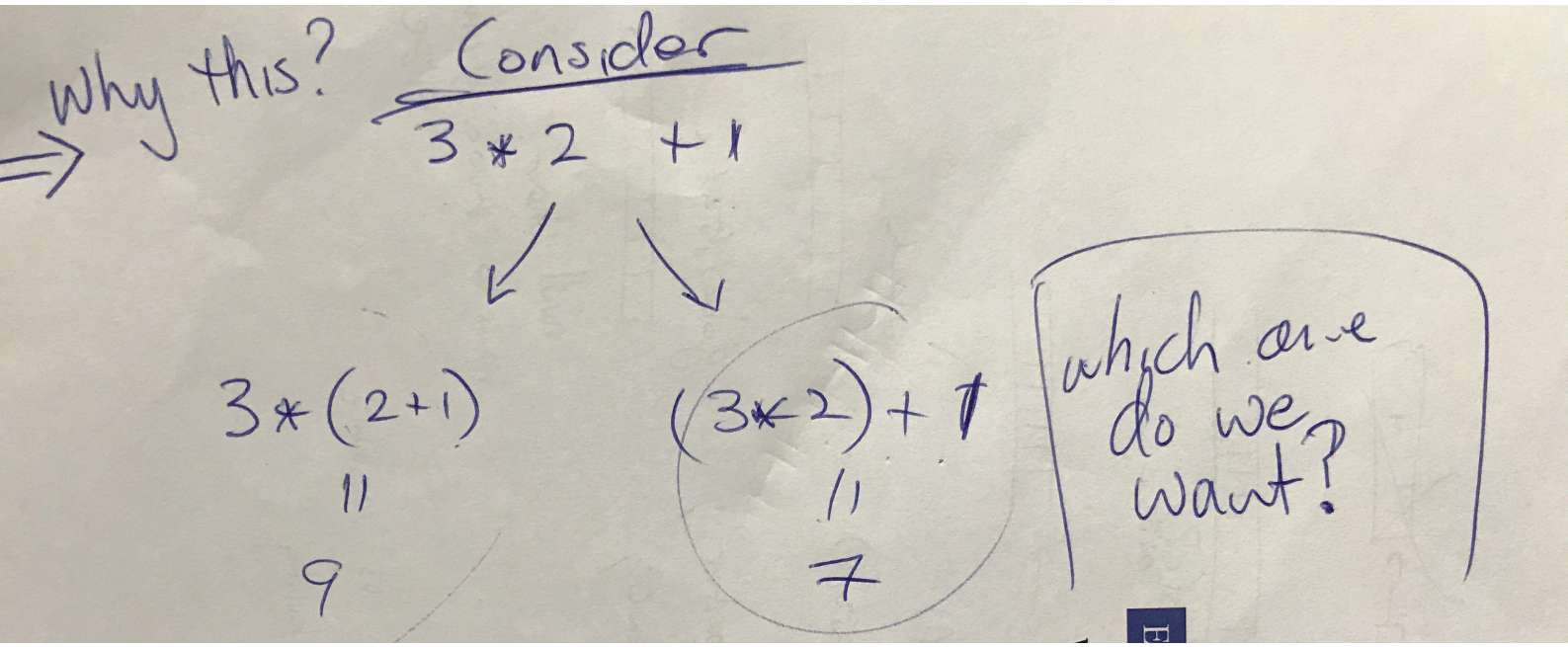
not (A and B)
↓
not A and not B

msg + 0 * sizeof(char)
↓
0

(0x1000)[0] → (0x1000 value of) → 'H'

&(0x1000 value of)
↓
0x1000

3 x × 1/x


Why this?
Consider
3 * 2 + 1
3*(2+1)
11
9
(3*2)+1
11
7
which one do we want?

```c
#include <stdio.h>
#include <string.h>

int main() {

    int numbers[5];

    char letters[7];

    char word_array[] = { 'P','O','I','N','T','E','R','S' };

    char *word = "POINTERS";

    char word_array_2[] = "POINTERS";

    char *word_array_3[] = { "POINTERS" };

    char **word_array_4[] = { &word, word_array_3 };

    letters[1] = word[3];

    numbers[4] = word[0];

    printf("numbers address: \t%p\n", numbers);
    printf("letters address: \t%p\n", letters);
    printf("word_array address: \t%p\n", word_array);
    printf("word address: \t\t%p\n", word);
    printf("word_array_2 address: \t%p\n", word_array_2);
    printf("word_array_3 address: \t%p\n", word_array_3);
    printf("word_array_4 address: \t%p\n", word_array_4);

    // how to print string using word_array_3?
    printf("word_array_3: %s\n", ???

    // how to print both strings using word_array_4?
    printf("word_array_4: %s %s\n", ???

    return 0;
}
```

9 bytes copied contiguously

0xA000

0x7500 | 0xA000 | 1 x 64 bits

***

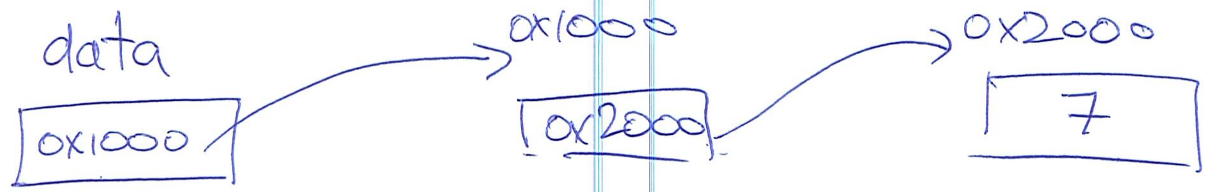
0x2500

0x7500

int **data

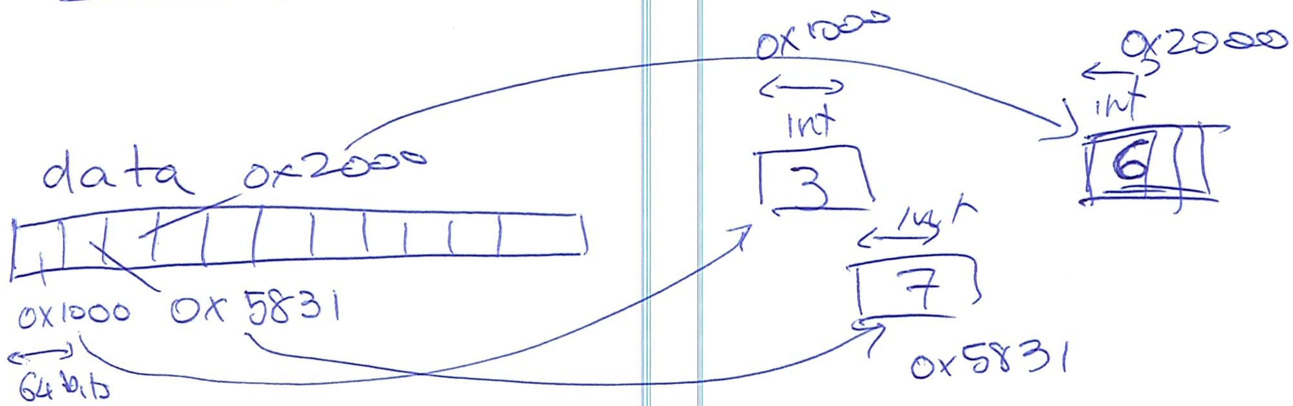
((*0x1000))

1

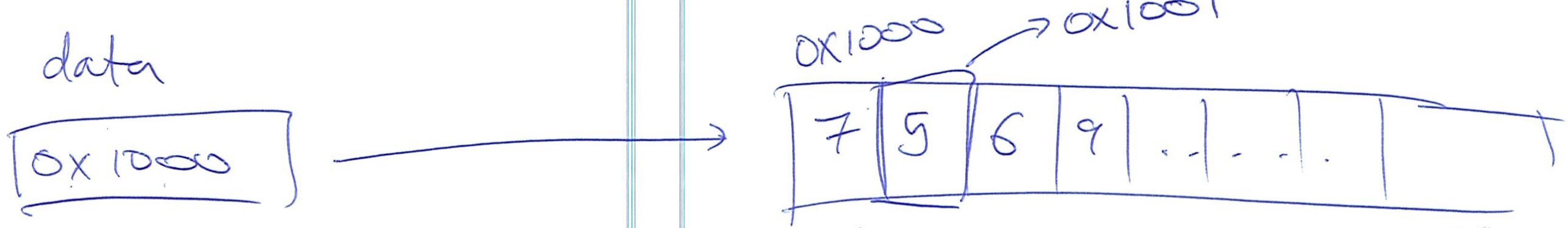


(*data)
↓
*(0x2000)
↓
7

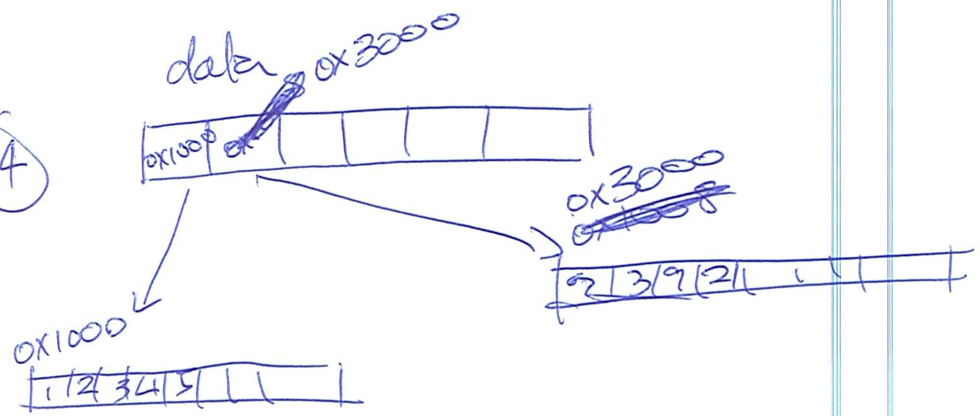
2



3

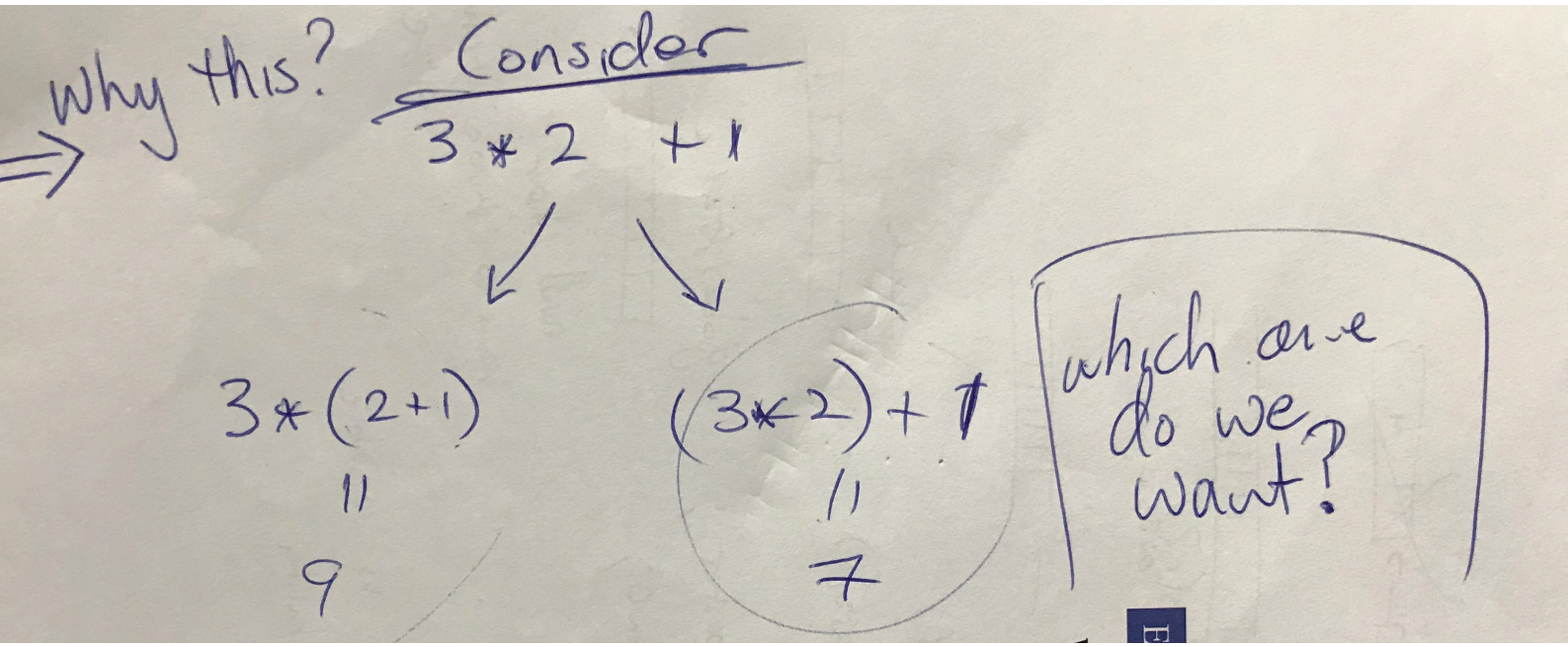


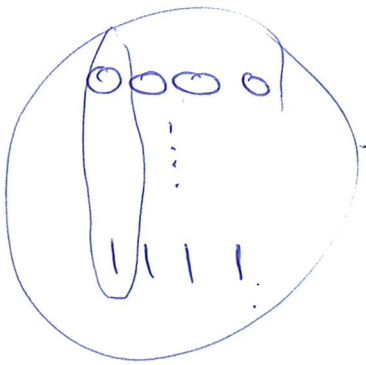
4



(*data)[1]
*((*data) + 1)
equivalent

*data == 0x1000
0x1000[1]


Why this?
Consider
3 * 2 + 1
3*(2+1)
11
9
(3*2)+1
11
7
which one do we want?

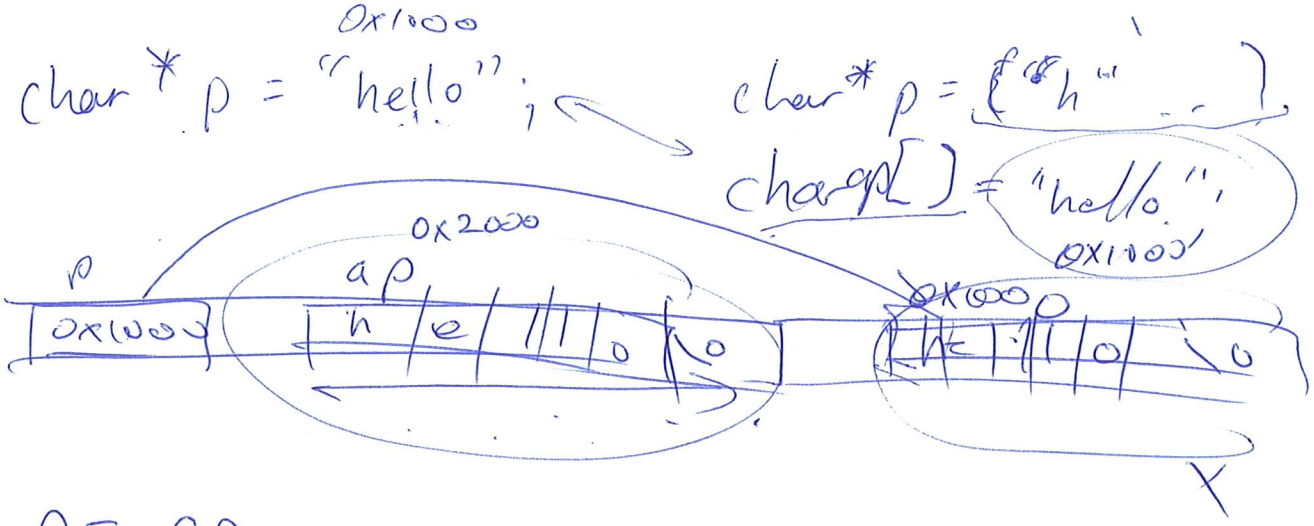


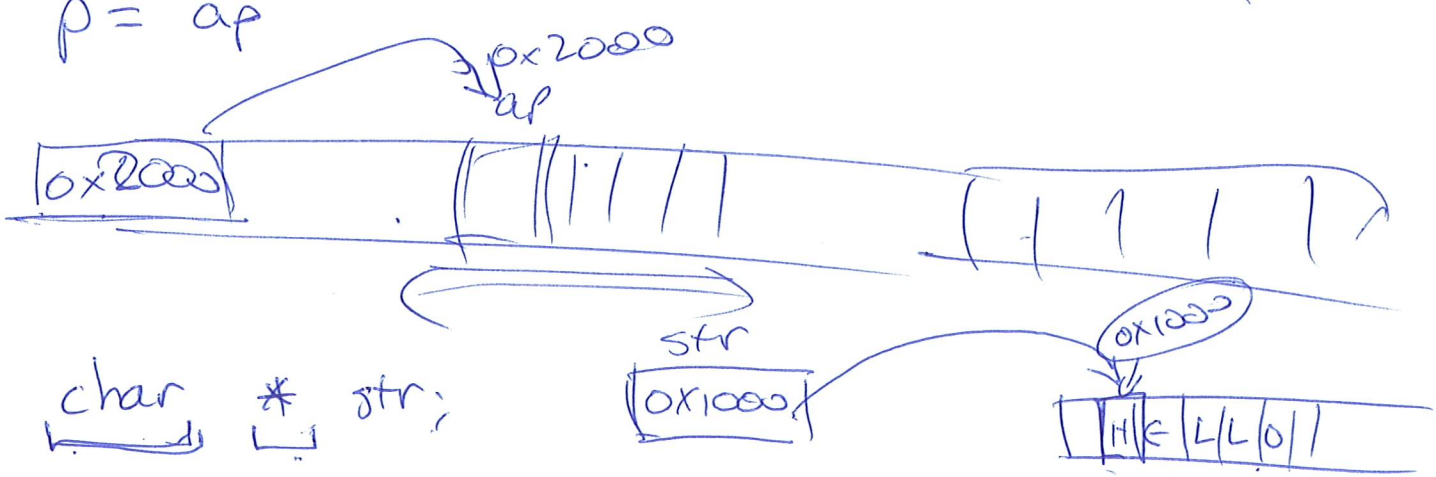
unsigned 0 → 15

signed −8 → 7

0 → 127

extended ASCII 0 − 255





<stdint.h> (uint32_t*) *str get sizeof(char) bytes

uint32_t *istr = "hello"; from address 0x1000

*istr → gets 4 bytes

print("%s\n", (char*) istr) → "hello"

~~char i = 0;~~
~~for(i<~~

uint32_t tmp = *istr;

~~char msg[] =~~

i = 0;
for(; i < 4; i++)
    printf("%c ", ((char*) tmp)[i]);
printf("\n");

0x1000

A = 65
a = 97

printf("%p\n")