BITS, BYTES, AND INTEGERS

CS 045

Computer Organization and Architecture

Prof. Donald J. Patterson

Adapted from Bryant and O'Hallaron,
Computer Systems:
A Programmer's Perspective, Third Edition

A FEW THOUGHTS

- A C program, once compiled, is a sequence of bytes
  - It has no information about the source code*

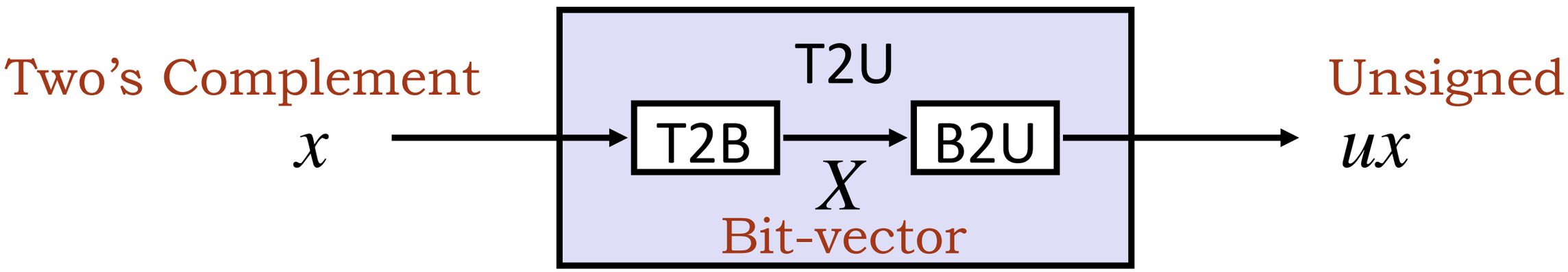
- A mask is a bit vector, that when &'d with a bit vector extracts a sub set of bits
  - mask = 0x70 = 0111 0000
  - source = 0xCA = 1100 1010 &
  - isolated bits = 0100 0000

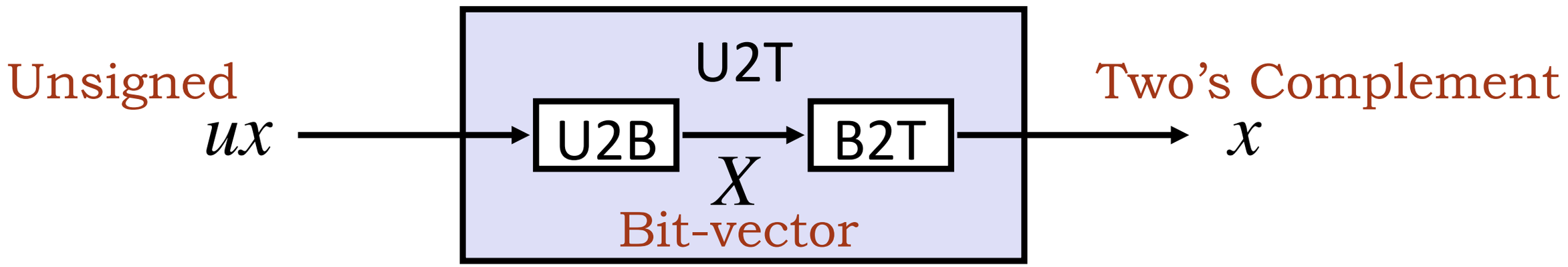
- For unsigned ints, >>, is logical
- For signed ints, >>, is undefined (but usually arithmetic)

CONVERSION AND CASTING

CONVERSION AND CASTING

MAPPING BETWEEN SIGNED AND UNSIGNED





- Keep bit representations and reinterpret

CONVERSION AND CASTING

MAPPING BETWEEN SIGNED AND UNSIGNED

Bits	Signed	Unsigned
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	-8	8
1001	-7	9
1010	-6	10
1011	-5	11
1100	-4	12
1101	-3	13
1110	-2	14
1111	-1	15

CONVERSION AND CASTING

MAPPING BETWEEN SIGNED AND UNSIGNED

Bits	Signed		Unsigned
0000	0		0
0001	1		1
0010	2		2
0011	3		3
0100	4		4
0101	5		5
0110	6		6
0111	7	T2U	7
1000	-8		8
1001	-7		9
1010	-6		10
1011	-5		11
1100	-4		12
1101	-3		13
1110	-2		14
1111	-1		15

CONVERSION AND CASTING

MAPPING BETWEEN SIGNED AND UNSIGNED

Bits	Signed		Unsigned
0000	0		0
0001	1		1
0010	2		2
0011	3		3
0100	4		4
0101	5		5
0110	6	→ T2U →	6
0111	7		7
1000	-8	← U2T ←	8
1001	-7		9
1010	-6		10
1011	-5		11
1100	-4		12
1101	-3		13
1110	-2		14
1111	-1		15

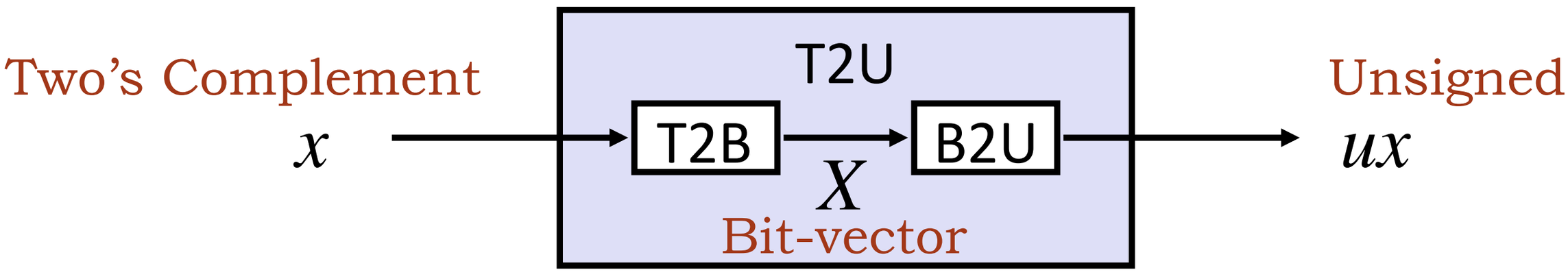
CONVERSION AND CASTING

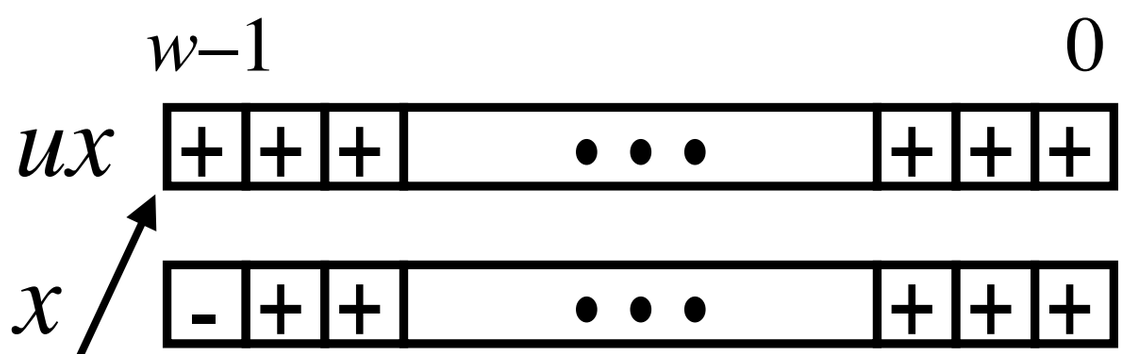
MAPPING BETWEEN SIGNED AND UNSIGNED

Bits	Signed		Unsigned
0000	0	=	0
0001	1	=	1
0010	2	=	2
0011	3	=	3
0100	4	=	4
0101	5	=	5
0110	6	=	6
0111	7	=	7
1000	-8	+/- 16	8
1001	-7	+/- 16	9
1010	-6	+/- 16	10
1011	-5	+/- 16	11
1100	-4	+/- 16	12
1101	-3	+/- 16	13
1110	-2	+/- 16	14
1111	-1	+/- 16	15

CONVERSION AND CASTING

MAPPING BETWEEN SIGNED AND UNSIGNED

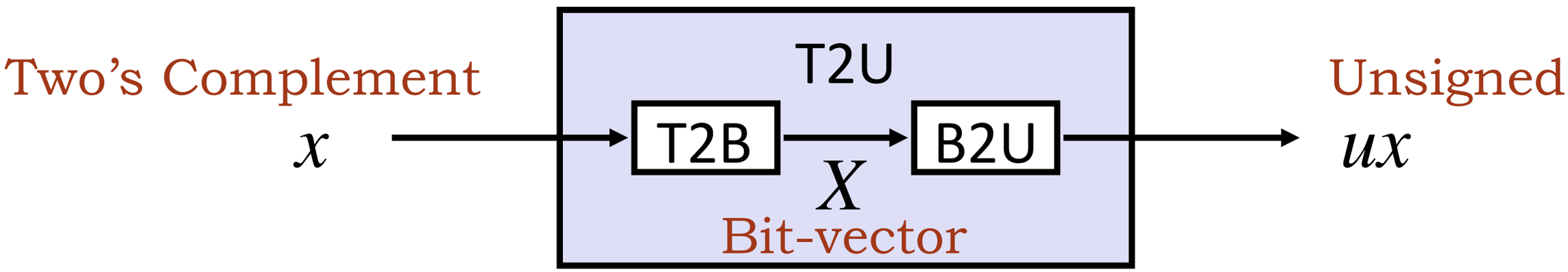


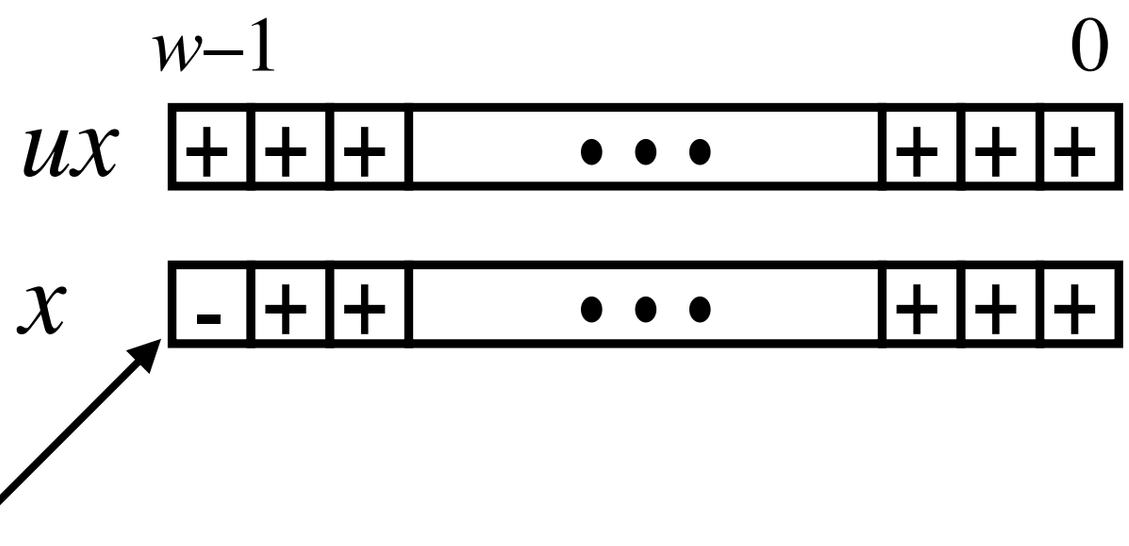


- Large positive weight

CONVERSION AND CASTING

MAPPING BETWEEN SIGNED AND UNSIGNED

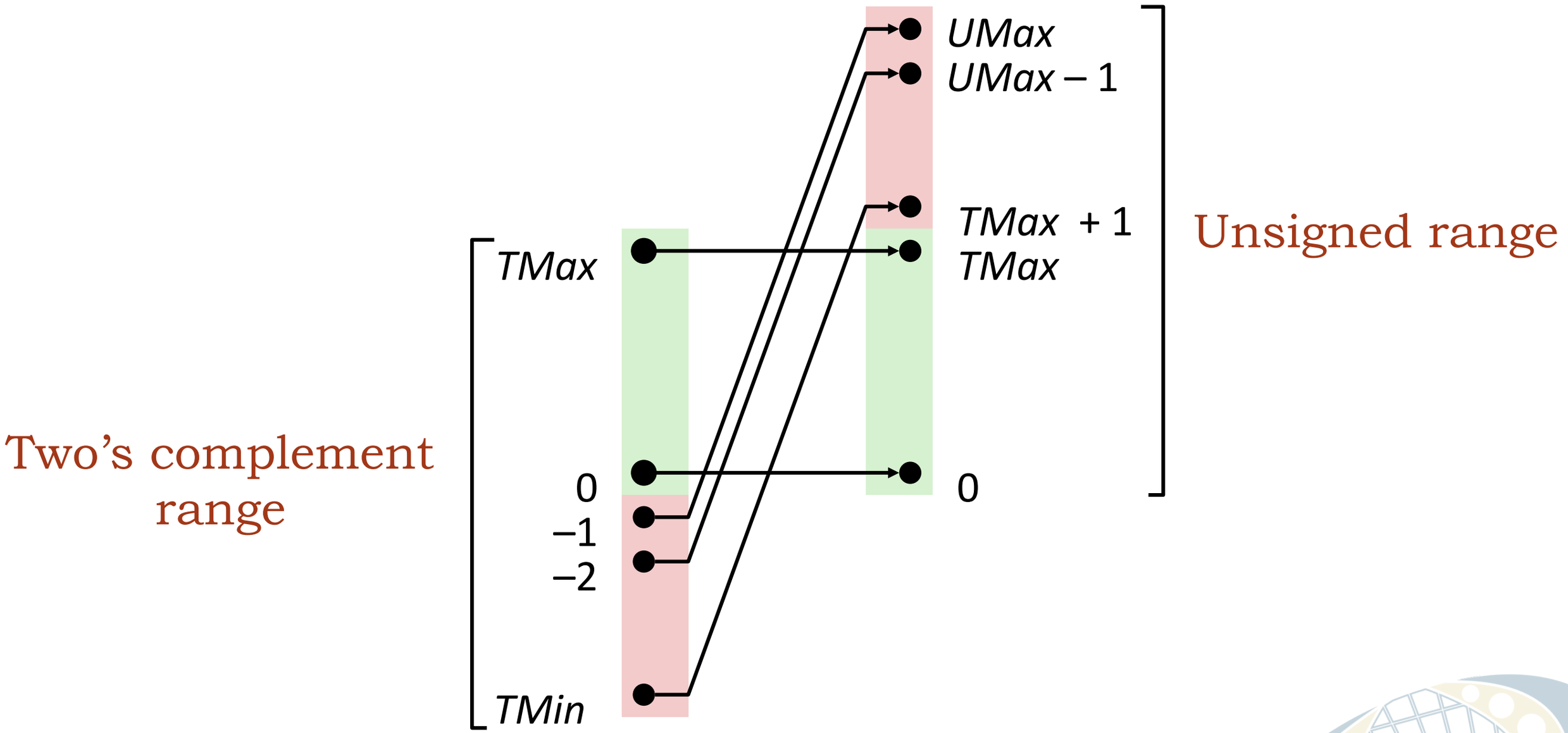




- Large negative weight

CONVERSION AND CASTING

MAPPING BETWEEN SIGNED AND UNSIGNED



CONVERSION AND CASTING

- Constants
  - By default are considered to be signed integers
  - Unsigned if have “u” as suffix
    - 0u, 4294967259u

CONVERSION AND CASTING

- Explicit Casting between signed & unsigned
  - same as U2T and T2U

```
int tx, ty;
unsigned ux, uy;

tx = (int) ux;
uy = (unsigned) ty;
```

- Implicit casting also occurs via assignments and procedure calls

```
int tx, ty;
unsigned ux, uy;

tx = ux; /* Cast to signed */
uy = ty; /* Cast to unsigned */
```

CONVERSION AND CASTING

- Meanwhile, printf doesn't care
  - printf doesn't respect types, it uses the types in the format tokens in the format string

```c
#include <stdio.h>

int main(){

    unsigned int a = 4294967295u;
    unsigned int b;
    signed int c = -1;
    signed int d;

    d = a; /* Cast to signed */
    b = c; /* Cast to unsigned */

    printf("a is %u %d 0x%x\n",a,a,a);
    printf("b is %u %d 0x%x\n",b,b,b);
    printf("c is %u %d 0x%x\n",c,c,c);
    printf("d is %u %d 0x%x\n",d,d,d);


    return 0;
}
```

CONVERSION AND CASTING

- Meanwhile, printf doesn't care
  - printf doesn't respect types, it uses the types in the format tokens in the format string

```c
#include <stdio.h>

int main(){

    unsigned int a = 4294967295u;
    unsigned int b;
    signed int c = -1;
    signed int d;

    d = a; /* Cast to signed */
    b = c; /* Cast to unsigned */

    printf("a is %u %d 0x%x\n",a,a,a);
    printf("b is %u %d 0x%x\n",b,b,b);
    printf("c is %u %d 0x%x\n",c,c,c);
    printf("d is %u %d 0x%x\n",d,d,d);


    return 0;
}
```

```
a is 4294967295 -1 0xffffffff
b is 4294967295 -1 0xffffffff
c is 4294967295 -1 0xffffffff
d is 4294967295 -1 0xffffffff
```

CONVERSION AND CASTING

GOTCHAS

- Expression Evaluation
  - If there is a mix of unsigned and signed in single expression, signed values implicitly cast to unsigned
    - Including comparison operations <, >, ==, <=, >=

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	**unsigned**
-1	0	<	**signed**
-1	0U	>	**unsigned**
2147483647	-2147483647-1	>	**signed**
2147483647U	-2147483647-1	<	**unsigned**
-1	-2	>	**signed**
(unsigned)-1	-2	>	**unsigned**
2147483647	2147483648U	<	**unsigned**
2147483647	(int) 2147483648U	>	**signed**

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	**unsigned**
-1	0	<	**signed**
-1	0U	>	**unsigned**
2147483647	-2147483647-1	>	**signed**
2147483647U	-2147483647-1	<	**unsigned**
-1	-2	>	**signed**
(unsigned)-1	-2	>	**unsigned**
2147483647	2147483648U	<	**unsigned**
2147483647	(int) 2147483648U	>	**signed**

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	**unsigned**
-1	0	<	**signed**
-1	0U	>	**unsigned**
2147483647	-2147483647-1	>	**signed**
2147483647U	-2147483647-1	<	**unsigned**
-1	-2	>	**signed**
(unsigned)-1	-2	>	**unsigned**
2147483647	2147483648U	<	**unsigned**
2147483647	(int) 2147483648U	>	**signed**

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	**unsigned**
-1	0	<	**signed**
-1	0U	>	**unsigned**
2147483647	-2147483647-1	>	**signed**
2147483647U	-2147483647-1	<	**unsigned**
-1	-2	>	**signed**
(unsigned)-1	-2	>	**unsigned**
2147483647	2147483648U	<	**unsigned**
2147483647	(int) 2147483648U	>	**signed**

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	unsigned
-1	0	<	signed
-1	0U	>	unsigned
2147483647	-2147483647-1	>	signed
2147483647U	-2147483647-1	<	unsigned
-1	-2	>	signed
(unsigned)-1	-2	>	unsigned
2147483647	2147483648U	<	unsigned
2147483647	(int) 2147483648U	>	signed

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	unsigned
-1	0	<	signed
-1	0U	>	unsigned
2147483647	-2147483647-1	>	signed
2147483647U	-2147483647-1	<	unsigned
-1	-2	>	signed
(unsigned)-1	-2	>	unsigned
2147483647	2147483648U	<	unsigned
2147483647	(int) 2147483648U	>	signed

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	**unsigned**
-1	0	<	**signed**
-1	0U	>	**unsigned**
2147483647	-2147483647-1	>	**signed**
2147483647U	-2147483647-1	<	**unsigned**
-1	-2	>	**signed**
(unsigned)-1	-2	>	**unsigned**
2147483647	2147483648U	<	**unsigned**
2147483647	(int) 2147483648U	>	**signed**

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	unsigned
-1	0	<	signed
-1	0U	>	unsigned
2147483647	-2147483647-1	>	signed
2147483647U	-2147483647-1	<	unsigned
-1	-2	>	signed
(unsigned)-1	-2	>	unsigned
2147483647	2147483648U	<	unsigned
2147483647	(int) 2147483648U	>	signed

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

0	0U	==	unsigned
-1	0	<	signed
-1	0U	>	unsigned
2147483647	-2147483647-1	>	signed
2147483647U	-2147483647-1	<	unsigned
-1	-2	>	signed
(unsigned)-1	-2	>	unsigned
2147483647	2147483648U	<	unsigned
2147483647	(int) 2147483648U	>	signed

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

```
( 0 == 0u ) is true
( -1 < 0 ) is true
( -1 < 0u ) is false
( 2147483647 > (-2147483647 - 1) ) is true
( 2147483647u > (-2147483647 - 1) ) is false
( -1 > -2 ) is true
( ((unsigned) -1) > -2 ) is true
( 2147483647 < 2147483648u ) is true
( 2147483647 < ((int)2147483648u) ) is false
```

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

```
( 0 == 0u ) is true
( -1 < 0 ) is true
( -1 < 0u ) is false
( 2147483647 > (-2147483647 - 1) ) is true
( 2147483647u > (-2147483647 - 1) ) is false
( -1 > -2 ) is true
( ((unsigned) -1) > -2 ) is true
( 2147483647 < 2147483648u ) is true
( 2147483647 < ((int)2147483648u) ) is false
```

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

```
( 0 == 0u ) is true
( -1 < 0 ) is true
( -1 < 0u ) is false★
( 2147483647 > (-2147483647 - 1) ) is true
( 2147483647u > (-2147483647 - 1) ) is false★
( -1 > -2 ) is true
( ((unsigned) -1) > -2 ) is true
( 2147483647 < 2147483648u ) is true
( 2147483647 < ((int)2147483648u) ) is false
```

CONVERSION AND CASTING

GOTCHAS

- Examples for W = 32
  - TMIN = -2,147,483,648 TMAX = 2,147,483,647

```
( 0 == 0u ) is true
( -1 < 0 ) is true
( -1 < 0u ) is false★
( 2147483647 > (-2147483647 - 1) ) is true
( 2147483647u > (-2147483647 - 1) ) is false★
( -1 > -2 ) is true
( ((unsigned) -1) > -2 ) is true
( 2147483647 < 2147483648u ) is true
( 2147483647 < ((int)2147483648u) ) is false★
```

CONVERSION AND CASTING

GOTCHAS

```c
#include <stdio.h>

float sum_elements(float a[], unsigned length){
    int i;
    float result = 0.0;

    for(i = 0 ; i <= length - 1 ; i++){
        result = result + a[i];
    }

    return result;
}

int main(){

    float x[10] = {0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0};

    printf("The sum of 10 elements is %f\n",sum_elements(x,10));
    printf("The sum of 1 elements is %f\n",sum_elements(x,1));
    printf("The sum of 0 elements is %f\n",sum_elements(x,0));

    return 0;
}
```

CONVERSION AND CASTING

GOTCHAS

```
The sum of 10 elements is 45.000000
The sum of 1 elements is 0.000000
Segmentation fault: 11
```

```c
#include <stdio.h>

float sum_elements(float a[], unsigned length){
    int i;
    float result = 0.0;

    for(i = 0 ; i <= length - 1 ; i++){
        result = result + a[i];
    }

    return result;
}

int main(){

    float x[10] = {0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0};

    printf("The sum of 10 elements is %f\n",sum_elements(x,10));
    printf("The sum of 1 elements is %f\n",sum_elements(x,1));
    printf("The sum of 0 elements is %f\n",sum_elements(x,0));

    return 0;
}
```

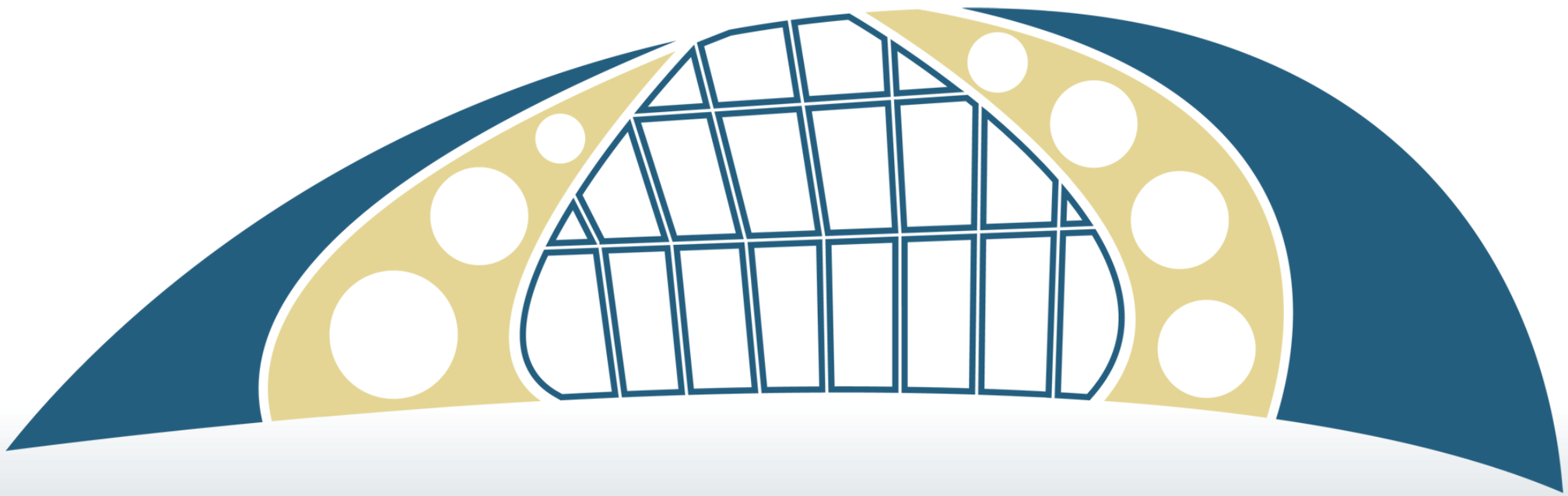
CONVERSION AND CASTING

SUMMARY
CASTING SIGNED ↔ UNSIGNED: BASIC RULES

- Bit pattern is maintained
- But reinterpreted
- Can have unexpected effects: adding or subtracting 2w

- Expression containing signed and unsigned int
  - `int` is cast to `unsigned`!!

- Rule of thumb:
  - if you are working with numbers use signed integers
  - if you are working with bit vectors use unsigned integers



WESTMONT INSPIRED

COMPUTING LAB