CSC 411

Computer Organization (Spring 2022)
Lecture 10: Examples

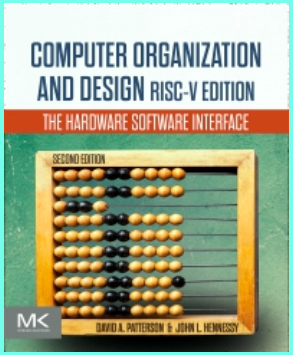
Prof. Marco Alvarez, University of Rhode Island

Disclaimer

Some of the following slides are adapted from:

Computer Organization and Design (Patterson and Hennessy)

The Hardware/Software Interface



Sum array example

```
int sum_array(int *p, int n) {

}

// arguments p in x10, n in x11
// return value in x10
```

What does this code do?

```
int foo(char *s) {
    int c = 0;
    while (*s++) {
        ++c;
    }
    return c;
}
```

Character data

- Byte encoded character sets
  - ASCII: 128 characters
    - 95 graphic, 33 control
  - Latin-1: 256 characters
    - ASCII, +96 more graphic characters
- Unicode: 32-bit character set
  - used in Java, C++ wide characters, …
  - most of the world's alphabets, plus symbols
  - UTF-8, UTF-16, UTF-32: variable-length encodings

ASCII representation

ASCII value	Char-acter	ASCII value	Char-acter	ASCII value	Char-acter	ASCII value	Char-acter	ASCII value	Char-acter	ASCII value	Char-acter
32	space	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

Byte/halfword/word operations

- Load byte/halfword/word
  - sign extend to 64 bits in `rd`
    - `lb rd, offset(rs1)`
    - `lh rd, offset(rs1)`
    - `lw rd, offset(rs1)`
- Load byte/halfword/word unsigned
  - zero extend to 64 bits in `rd`
    - `lbu rd, offset(rs1)`
    - `lhu rd, offset(rs1)`
    - `lwu rd, offset(rs1)`
- Store byte/halfword/word
  - store rightmost 8/16/32 bits
    - `sb rs2, offset(rs1)`
    - `sh rs2, offset(rs1)`
    - `sw rs2, offset(rs1)`

String copy example

```
void strcpy (char x[], char y[]) {
    int i = 0;
    while ((x[i]=y[i]) != '\0') {
        i += 1;
    }
}
```

RISC-V code

```c
// addresses of x, y in x10, x11
// i in x19
void strcpy (char x[], char y[]) {
    int i = 0;
    while ((x[i]=y[i]) != '\0')
        i += 1;
}
```

```
strcpy:
    addi sp, sp, -8    // adjust stack for 1 doubleword
    sd   x19, 0(sp)    // push x19
    add  x19, x0, x0   // i=0
L1:
    add  x5, x19, x11  // x5 = addr of y[i]
    lbu  x6, 0(x5)     // x6 = y[i]
    add  x7, x19, x10  // x7 = addr of x[i]
    sb   x6, 0(x7)     // x[i] = y[i]
    beq  x6, x0, L2    // if y[i] == 0 then exit
    addi x19, x19, 1   // i = i + 1
    jal  x0, L1        // next iteration of loop
L2:
    ld   x19, 0(sp)    // restore saved x19
    addi sp, sp, 8     // pop 1 doubleword from stack
    jalr x0, 0(x1)     // and return
```