More Data Representation (filling in some gaps)

January 18, 2013

Outline

Some more comments on integers

- Carry out vs. overflow
- Identifying negative integers

Representing characters and strings

- ASCII encoding
- Strings in MIPS and endianness

Carry out vs. overflow

Carry out: carry after most significant bit $\Rightarrow$ discard, no error
Overflow: result is out of representable range $\Rightarrow$ error!

Carry out $\neq$ overflow!

Carry out is a normal part of signed integer addition

Will get a carry out when adding:

- two negative numbers
- a negative and a positive, result is positive

Just ignore it!

Identifying negative integers, binary

Assume 16-bit, byte-addressable signed integers

Big endian

- 0101 1010 0000 1100 $\Rightarrow$ positive
- 1000 0110 1001 0101 $\Rightarrow$ negative

Little endian

- 0000 1100 0101 1010 $\Rightarrow$ positive
- 1001 0101 1000 0110 $\Rightarrow$ negative

Sign determined by most significant bit

Identifying negative numbers, hex

Can you easily tell if a signed integer written in hex is negative?

- if most significant digit is 0–7 $\Rightarrow$ positive
- if most significant digit is 8–F $\Rightarrow$ negative

Big endian

- `0x5A0C` $\Rightarrow$ positive
- `0x8695` $\Rightarrow$ negative
- `0x0B30` $\Rightarrow$ positive
- `0xC110` $\Rightarrow$ negative

Little endian

- `0x0C5A` $\Rightarrow$ positive
- `0x9586` $\Rightarrow$ negative
- `0x300B` $\Rightarrow$ positive
- `0x10C1` $\Rightarrow$ negative

Outline

Representing characters

What is a character?

- letter, digit, symbols, newline, null, . . .
- all keyboard input (even “numbers”)

Like all data, characters are encoded as binary numbers

Subset of ASCII encoding (7 bits, `0x00`–`0x7F`):

Character(s)	Hex representation
‘\0’	`0x00`
‘\n’	`0x0A`
‘0’–‘9’	`0x30`–`0x39`
‘A’–‘Z’	`0x41`–`0x5A`
‘a’–‘z’	`0x61`–`0x7A`

ASCII encoding

Character(s)	Hex representation
'\0'	`0x00`
'\n'	`0x0A`
'0'–'9'	`0x30`–`0x39`
'A'–'Z'	`0x41`–`0x5A`
'a'–'z'	`0x61`–`0x7A`

Note that '1' = `0x31`, not `0x01`!

Some patterns we can exploit:

- uppercase to lowercase: add `0x20`
- lowercase to uppercase: subtract `0x20`
- character digit to numeric value: subtract `0x30`

Complete ASCII table

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

To encode character: **0x**(row)(col)

Examples

‘Q’ $\Rightarrow$ **0x51** ‘k’ $\Rightarrow$ **0x6B** ‘?’ $\Rightarrow$ **0x3F**

Representing strings

What is a string? In MIPS:

- sequence of ASCII-encoded characters (padded to 8-bits)
- ends in '\0' (null-terminated)
- padded with `0x00` bytes to make an even number of words

Examples (big endian)

- "Cat" $\Rightarrow$
  `0x43617400`
- "Cats?" $\Rightarrow$
  `0x43617473 3F000000`

Character(s)	Hex
'\0'	`0x00`
'\n'	`0x0A`
'0'–'9'	`0x30`–`0x39`
'A'–'Z'	`0x41`–`0x5A`
'a'–'z'	`0x61`–`0x7A`

Numbers as strings

When you type a number as input to a program, it is a string!

- if you want to use it as a number, you must convert it
- we'll do this later in the course

Examples

- "512" $\Rightarrow$ `0x35313200`
- "1024" $\Rightarrow$ `0x31303234 00000000`

Little-endian strings in MIPS

Endianness refers only to the order of bytes **within a word**

- makes multi-word little-endian strings confusing . . .

Little-endian strings

- order of words is same as big endian
- order of bytes within words is reversed

String	“Cat”	“Cats?”
Big endian	`0x43617400`	`0x43617473 3F000000`
Little endian	`0x00746143`	`0x73746143 0000003F`

In-class exercises (part 1)

Assume 32-bit (byte addressable) signed integers

For each number, is it positive or negative . . .

- if big-endian?
- if little-endian?

Numbers

1. `0x12345678`
2. `0x456789AB`
3. `0xCAFEBABE`
4. `0xCAFED00D`

In-class exercises (part 2)

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

In hex, write the string “Kaplow” as a big-endian and a little-endian, null-terminated ASCII string, as in MIPS

- **`0x4B61706C 6F770000`**
- **`0x6C70614B 0000776F`**