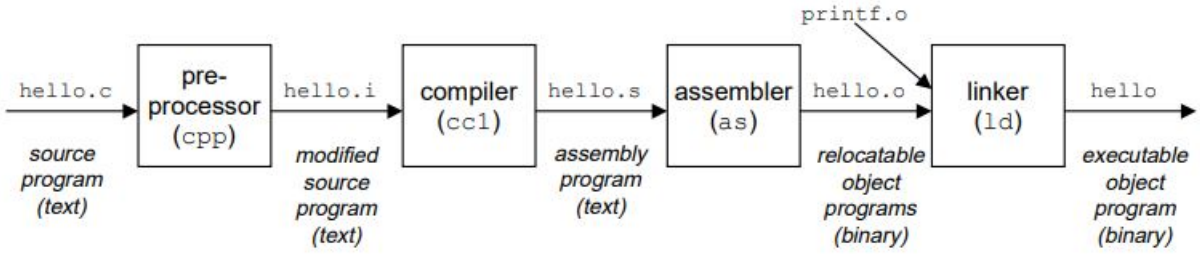
서강대학교 Sogang University

Introduction to Computer Systems

Ex. 1 – Pen and Paper (Chapter 01 + 02)

Chapter 1.2 : Program translation process

The following figure shows the translation of a high-level C program into binary code, that is read and operated by the processor. Answer the following questions.

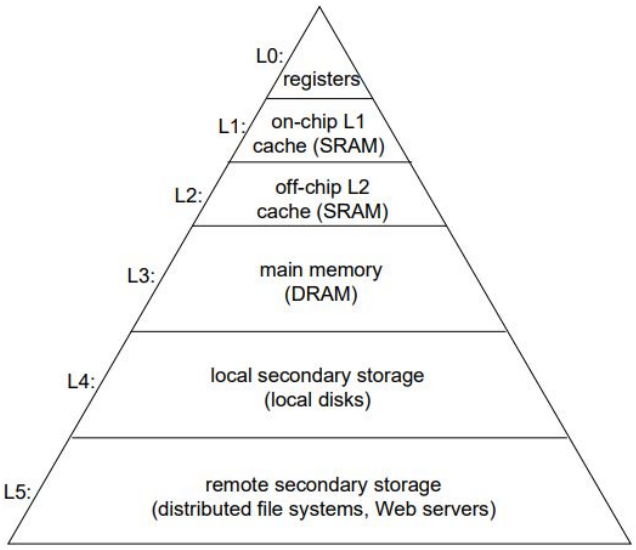


a) Explain each process of the translation.

b) Why do application program developers need high-level programming languages and compilers rather than assembly language or machine language?

Chapter 1.6 : Storage Devices and its hierarchy

The storage devices in every computer system are organized as the memory hierarchy shown in the following figure. As we move from the top of the hierarchy to the bottom, devices become slower and larger. (For example, the disk drive on a typical system might be 100 times larger than the main memory, but it might take for the processor 10,000,000 times longer to read a word from disk than from memory.) Answer the following questions.



a) Explain the role of the registers.

b) Explain the role of the main memory.

c) Caches are located between main memory and registers. What is the advantage of using caches?

Chapter 2.1 : Hexadecimal Notation

Convert given decimal numbers to an 8-bit unsigned binary numbers and corresponding hexadecimal numbers. If not possible, write X as your answer. (e.g., 87→01010111, 57)

a) 0

b) 255

c) 256

d) 100

Chapter 2.1 : Addressing and Byte Ordering

Suppose that variables x and y have type *int* and *short* respectively. x has address 0x100, and y has address 0x200. Then four bytes of x would be stored in memory locations 0x100, 0x101, 0x102, and 0x103. Similarly, two bytes of y would be stored in memory locations 0x200 and 0x201. The value of x is 27066166, and 41797 for y. If we store x and y in each *little endian* and *big endian*, what values are stored in the given memory range? (represent as hexadecimal.)

Little endian :

Address	Value
0x100	
0x101	
0x102	
0x103	
...	
0x200	
0x201	

Big endian :

Address	Value
0x100	
0x101	
0x102	
0x103	
...	
0x200	
0x201	

Chapter 2.1 : Application of bit vectors

A w-bit vector can represent a subset of $\{0, \ldots, w–1\}$. Represent each given two unsigned integers in 8-bit binary code, which means an 8-bit vector, and then calculate the intersection, union, and symmetric difference of the two sets. The elements of the set are $\{0, \ldots, 7\}$. Each element corresponds in ascending order from the left bit.

a) 28, 64

b) 63, 112

Chapter 2.1, 2.2 : Strings and Integers

A given 32-bit binary code represents strings or integers (using two's complement encoding). If the hexadecimal representation of each byte is all in the conversion table below, the given code is a string; otherwise it is an integer. Write down the string or integer for each binary code. (from left to right)
Example:

Binary code	01100101 01110011 01101110 01100111
Hexadecimal representation	65 73 6E 67
Character representation	esng

Dec	Oct	Hex	C	Dec	Oct	Hex	C	Dec	Oct	Hex	C	Dec	Oct	Hex	C
0	0	0	^@	32	40	20		64	100	40	@	96	140	60	`
1	1	1	^A	33	41	21	!	65	101	41	A	97	141	61	a
2	2	2	^B	34	42	22	"	66	102	42	B	98	142	62	b
3	3	3	^C	35	43	23	#	67	103	43	C	99	143	63	c
4	4	4	^D	36	44	24	$	68	104	44	D	100	144	64	d
5	5	5	^E	37	45	25	%	69	105	45	E	101	145	65	e
6	6	6	^F	38	46	26	&	70	106	46	F	102	146	66	f
7	7	7	^G	39	47	27	'	71	107	47	G	103	147	67	g
8	10	8	^H	40	50	28	(	72	110	48	H	104	150	68	h
9	11	9	^I	41	51	29	)	73	111	49	I	105	151	69	i
10	12	a	^J	42	52	2a	*	74	112	4a	J	106	152	6a	j
11	13	b	^K	43	53	2b	+	75	113	4b	K	107	153	6b	k
12	14	c	^L	44	54	2c	,	76	114	4c	L	108	154	6c	l
13	15	d	^M	45	55	2d	-	77	115	4d	M	109	155	6d	m
14	16	e	^N	46	56	2e	.	78	116	4e	N	110	156	6e	n
15	17	f	^O	47	57	2f	/	79	117	4f	O	111	157	6f	o
16	20	10	^P	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	^Q	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	^R	50	62	32	2	82	122	52	R	114	162	72	r
19	23	13	^S	51	63	33	3	83	123	53	S	115	163	73	s
20	24	14	^T	52	64	34	4	84	124	54	T	116	164	74	t
21	25	15	^U	53	65	35	5	85	125	55	U	117	165	75	u
22	26	16	^V	54	66	36	6	86	126	56	V	118	166	76	v
23	27	17	^W	55	67	37	7	87	127	57	W	119	167	77	w
24	30	18	^X	56	70	38	8	88	130	58	X	120	170	78	x
25	31	19	^Y	57	71	39	9	89	131	59	Y	121	171	79	y
26	32	1a	^Z	58	72	3a	:	90	132	5a	Z	122	172	7a	z
27	33	1b	^[	59	73	3b	;	91	133	5b	[	123	173	7b	{
28	34	1c	^\	60	74	3c	<	92	134	5c	\	124	174	7c	|
29	35	1d	^]	61	75	3d	=	93	135	5d	]	125	175	7d	}
30	36	1e	^^	62	76	3e	>	94	136	5e	^	126	176	7e	~
31	37	1f	^_	63	77	3f	?	95	137	5f	_	127	177	7f	

a) 01000101 01010011 01001110 01000111

b) 01110100 00100001 00110011 00101110

c) 11111111 11111111 11111111 11010100

d) 00000000 00000001 01010111 10100110

Chapter 2.3 : Integer Arithmetic

Answer the following questions.

a) A data type uses unsigned integer representation and it is defined on x bits. When we add two numbers 211 and 240 in this data type, it do not cause overflow. However, when we add two numbers 384 and 247 in this data type, it causes overflow. What is a value of x?

b) A data type is defined on 4 bits. Let x be a variable of this data type. Here, we want calculate $3x$ without overflow. If this data type uses unsigned integer representation, what is the range of x? Also, if it uses signed integer representation with two's complement encoding, what is the range of x?

Chapter 2.3 Two's Complement Multiplication

The following table shows integer multiplication of 3-bit unsigned and two's complement encoded bits. A hardware that performs this operation leaves only three bits on the right side of the multiplication result. Fill in the blanks in the table.

Mode	x	y	$x \cdot y$ (int)	$x \cdot y$ (binary)	Truncated $x \cdot y$ (binary)	Truncated $x \cdot y$ (int)	Sign of the truncated result
Unsigned	5	3	15	001111	111	7	+
Two's Comp.	-3	3	-9	110111	111	-1	-
Unsigned	3	2					
Two's Comp.	3	2					
Unsigned	3	7					
Two's Comp.	-4	3					
Unsigned	7	1					
Two's Comp.	-1	1					

Chapter 2.4 : Fractional Binary Numbers

Convert a given fractional number or its binary or decimal representation to other two representations.

Fractional number	Binary representation	Decimal representation
77/32		
	0.0111	
		89.125

Chapter 2.4 : An example for the imprecision of floating-point arithmetic

The fractional binary representation for 0.1 is 0.000110011[0011]... where [0011] means a non-terminating repeated sequence. Suppose a computer approximates this value with the leading bit (integer bit) plus the first 13 bits of this sequence. Let this value $x = 0.0001100110011$. Answer the following question.

a) What is the fractional binary representation of $7 - x$?

b) What is the decimal representation of $7 - x$?

c) Compare the value of $10^{13}(7 - 0.1)$ with $10^{13}(7 - x)$.

Chapter 2.4 : IEEE Floating-Point Representation

A 9-bit float-point representation format has s = 1 sign bit, k = 4 exponent bits, and n = 4 significand bits. Then, the *bias* is $2^{4-1} - 1 = 7$. Assume that the exponent field representing the value e is not all 1s. The fraction field represents a value f. Answer the following questions.

a) For *normalized values*, how to compute exponent E? How to compute significand M?

b) What is a possible numeric range for E = -4 ?

c) For *denormalized values*, how to compute exponent E?

d) What is a possible numeric range for *denormalized values*?