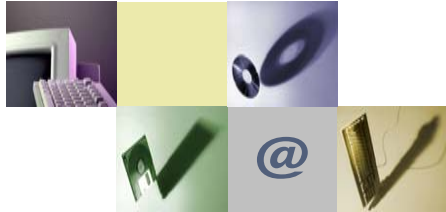


Computer Architecture (S/W)



Digital Computer Concept and Practice

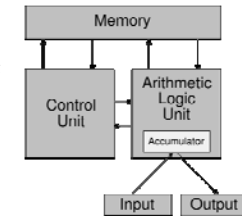
Contents

- Computer Operation
- Data Representation
- Instruction Representation

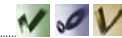
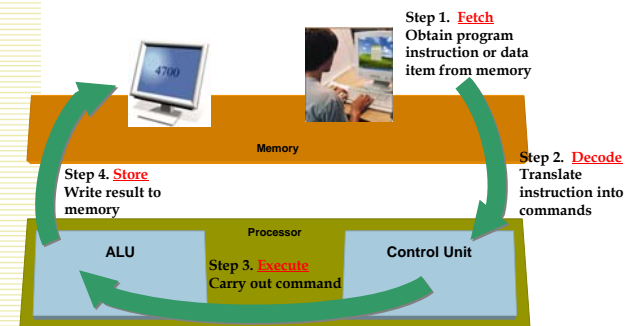


CPU Process

- Von Neumann architecture
 - Processing unit
 - A class of logic machines that can execute computer programs
 - Storage unit
 - Hold data, instructions and programs
 - Execute programs
 - A CPU reads stored instructions in order from storage units and decodes and executes actions that they indicate.



Process Cycle



Pipelining

- Pipelining is an implementation technique in which multiple instructions are overlapped in execution.
- CPU begins fetching second instruction before completing machine cycle for first instruction.
- A sort of parallel processing
- Results in faster processing

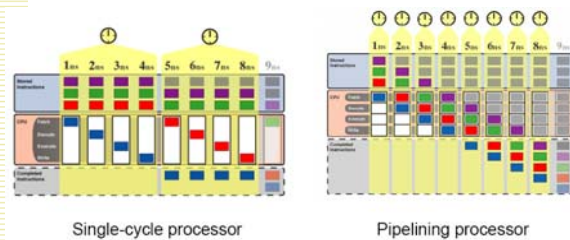


Storage Unit (Memory)

	Address	Storage values
Program	...	
	16	Store 3 in the variable X
	20	Store 1 in the variable Y
	24	Store X + Y in the variable Z
Data	...	
	32	X : 3
	36	Y : 1
	40	Z : 4



Single-Cycle Processor vs. Pipelining

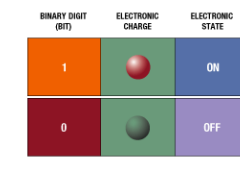


- For single-cycle processor it takes 16 nanosecond to execute four instructions, while for pipelining processor it takes only 7 nanoseconds.



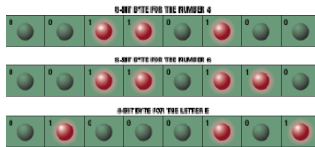
Data Representation

- How do computers represent data?
 - Digital
 - Recognize only two electronic states: on or off
 - Use a binary number system to recognize two states
 - 0 and 1, called bits (short for binary digits)
 - Word: data processing unit in the computer



Data Representation

- Byte
 - Eight bits grouped together as a unit
 - Provides enough different combinations of 0s and 1s to represent 256 individual characters
 - 8 bits = 1 byte $\rightarrow 2^8=256$ information
 - Numbers, uppercase and lowercase letters, punctuation marks etc.



Data Representation

- Types of coding systems to represent data
 - ASCII – American Standard Code for Information Interchange
 - It provides for 256 characters
 - E.g.) 01000001 – A, 01000010 – B
 - EBCDIC – Extended Binary Coded Decimal Interchange Code
 - Unicode – coding scheme capable of representing all world's languages
 - UTF (Unicode Transformation Format) encodings
 - UCS (Universal Character Set) encodings



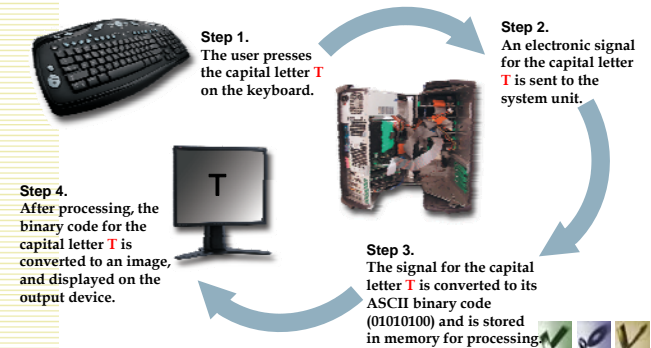
ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Html	Chr	Dec	Hex	Oct	Html	Chr	Dec	Hex	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	␣	64	40	100	␣	␣	96	60	140	␣	␣
1	1	001	SOH	(start of heading)	33	21	041	␣	65	41	101	␣	␣	97	61	141	␣	␣
2	2	002	STX	(start of text)	34	22	042	␣	66	42	102	␣	␣	98	62	142	␣	␣
3	3	003	ETX	(end of text)	35	23	043	␣	67	43	103	␣	␣	99	63	143	␣	␣
4	4	004	EOT	(end of transmission)	36	24	044	␣	68	44	104	␣	␣	100	64	144	␣	␣
5	5	005	ENQ	(enquiry)	37	25	045	␣	69	45	105	␣	␣	101	65	145	␣	␣
6	6	006	ACK	(acknowledge)	38	26	046	␣	70	46	106	␣	␣	102	66	146	␣	␣
7	7	007	BEL	(bell)	39	27	047	␣	71	47	107	␣	␣	103	67	147	␣	␣
8	8	010	BS	(backspace)	40	28	050	␣	72	48	110	␣	␣	104	68	150	␣	␣
9	9	011	TAB	(horizontal tab)	41	29	051	␣	73	49	111	␣	␣	105	69	151	␣	␣
10	A	012	LF	(NL line feed, new line)	42	2A	052	␣	74	4A	112	␣	␣	106	6A	152	␣	␣
11	B	013	VT	(vertical tab)	43	2B	053	␣	75	4B	113	␣	␣	107	6B	153	␣	␣
12	C	014	FF	(NP form feed, new page)	44	2C	054	␣	76	4C	114	␣	␣	108	6C	154	␣	␣
13	D	015	CR	(carriage return)	45	2D	055	␣	77	4D	115	␣	␣	109	6D	155	␣	␣
14	E	016	SO	(shift out)	46	2E	056	␣	78	4E	116	␣	␣	110	6E	156	␣	␣
15	F	017	SI	(shift in)	47	2F	057	␣	79	4F	117	␣	␣	111	6F	157	␣	␣
16	10	020	DLE	(data link escape)	48	30	060	␣	80	50	120	␣	␣	112	70	160	␣	␣
17	11	021	DC1	(device control 1)	49	31	061	␣	81	51	121	␣	␣	113	71	161	␣	␣
18	12	022	DC2	(device control 2)	50	32	062	␣	82	52	122	␣	␣	114	72	162	␣	␣
19	13	023	DC3	(device control 3)	51	33	063	␣	83	53	123	␣	␣	115	73	163	␣	␣
20	14	024	DC4	(device control 4)	52	34	064	␣	84	54	124	␣	␣	116	74	164	␣	␣
21	15	025	NAK	(negative acknowledge)	53	35	065	␣	85	55	125	␣	␣	117	75	165	␣	␣
22	16	026	SYN	(synchronous idle)	54	36	066	␣	86	56	126	␣	␣	118	76	166	␣	␣
23	17	027	ETB	(end of trans. block)	55	37	067	␣	87	57	127	␣	␣	119	77	167	␣	␣
24	18	030	CAN	(cancel)	56	38	070	␣	88	58	130	␣	␣	120	78	170	␣	␣
25	19	031	EM	(end of medium)	57	39	071	␣	89	59	131	␣	␣	121	79	171	␣	␣
26	1A	032	SUB	(substitute)	58	3A	072	␣	90	5A	132	␣	␣	122	7A	172	␣	␣
27	1B	033	ESC	(escape)	59	3B	073	␣	91	5B	133	␣	␣	123	7B	173	␣	␣
28	1C	034	FS	(file separator)	60	3C	074	␣	92	5C	134	␣	␣	124	7C	174	␣	␣
29	1D	035	GS	(group separator)	61	3D	075	␣	93	5D	135	␣	␣	125	7D	175	␣	␣
30	1E	036	RS	(record separator)	62	3E	076	␣	94	5E	136	␣	␣	126	7E	176	␣	␣
31	1F	037	US	(unit separator)	63	3F	077	␣	95	5F	137	␣	␣	127	7F	177	␣	␣

Source: www.LookupTables.com

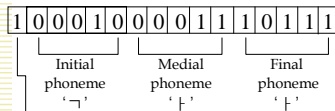
Data Representation Process

- How is a letter converted to binary form and back?



Hangul Character Representation

- Combination code (조합형)
 - The beginning consonant, a middle vowel and an optional ending consonant are represented by each 5 bit.
- Completion code (완성형)
 - A pre-defined set of Korean character codes, which maps pre-selected Hangul into two-byte coding space



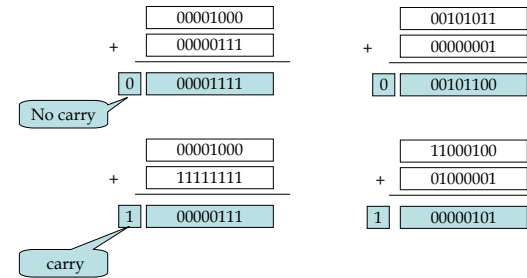
English = 0
Hangul = 1

Number (Hexadecimal)	Character
B0A1	가
B0A2	각
B0A3	간
...	...

- Currently, Unicode is the standard



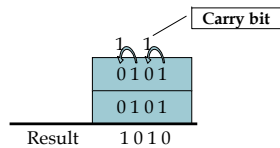
Binary Arithmetic Example



Binary Arithmetic

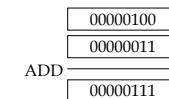
- The rules for binary arithmetic are:

0 + 0 = 0, carry = 0
1 + 0 = 1, carry = 0
0 + 1 = 1, carry = 0
1 + 1 = 0, carry = 1

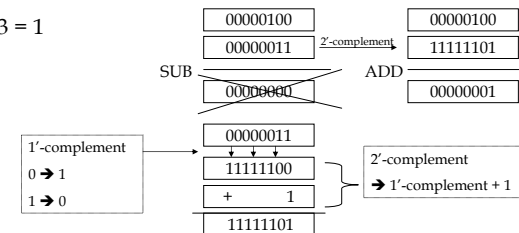


Handling Numbers (Integers)

- 4 + 3 = 7



- 4 - 3 = 1



Logical Operation

- AND

Input 1	input2	result
0	0	0
0	1	0
1	0	0
1	1	1

AND	<table border="1"><tr><td>01010111</td></tr><tr><td>10010001</td></tr><tr><td>00010001</td></tr></table>	01010111	10010001	00010001
01010111				
10010001				
00010001				
- OR

input1	input2	result
0	0	0
0	1	1
1	0	1
1	1	1

OR	<table border="1"><tr><td>01010111</td></tr><tr><td>10010001</td></tr><tr><td>11010111</td></tr></table>	01010111	10010001	11010111
01010111				
10010001				
11010111				
- NOT

input	result
0	1
1	0

NOT	<table border="1"><tr><td>10010001</td></tr><tr><td>01101110</td></tr></table>	10010001	01101110
10010001			
01101110			
- XOR



Instruction Representation

- Instructions are coded as a sequence of binary digits.
- Instruction: operation code + address
- E.g.) $A1 \leftarrow A2 + A3$
 - Load A2, 3: (hexadecimal) 05 02 00 03
 - Load A3, 1: 05 03 00 01
 - ADD A1, A2, A3: 70 01 02 03

