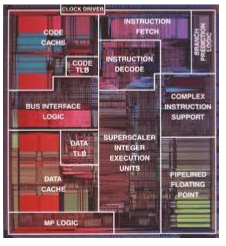
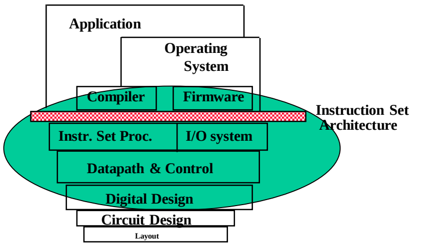
UTD

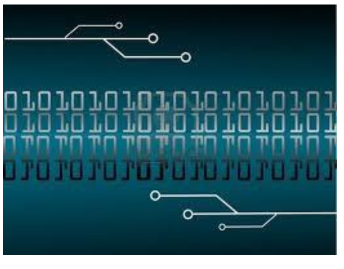
CS/SE 3340
Computer Architecture

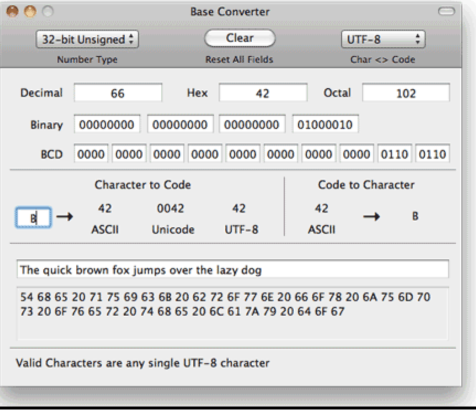




Data Representation & Base Conversion

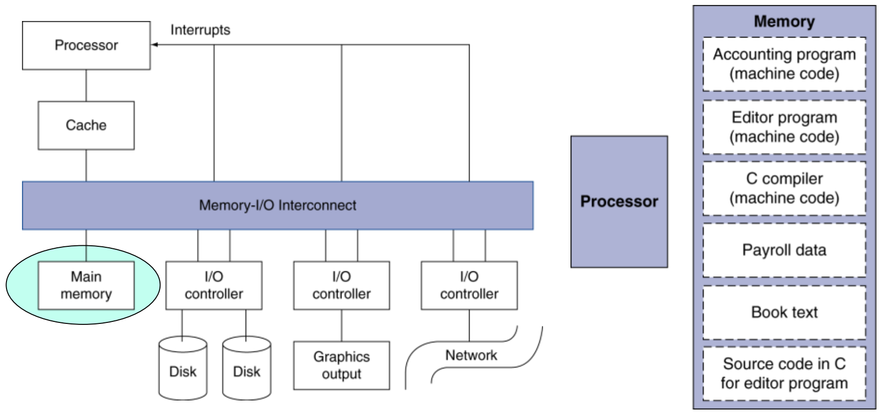
Adapted from "Computer Organization and Design, 4th Ed." by D. Patterson and J. Hennessy





2

Stored Program Computer



*Both **programs** (instructions) and **data** are stored in main memory*

3

Memory Organization

- Main memory can be viewed as a large one-dimension array of bytes (or octets)
- A memory address is an *index* into the memory array
- *Byte addressing* means that the index points to a byte of memory

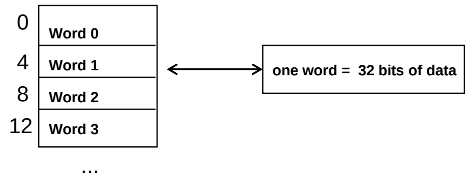
0	8 bits of data
1	8 bits of data
2	8 bits of data
3	8 bits of data
4	8 bits of data
5	8 bits of data
6	8 bits of data

...

But what do we do if our data is larger than a byte?

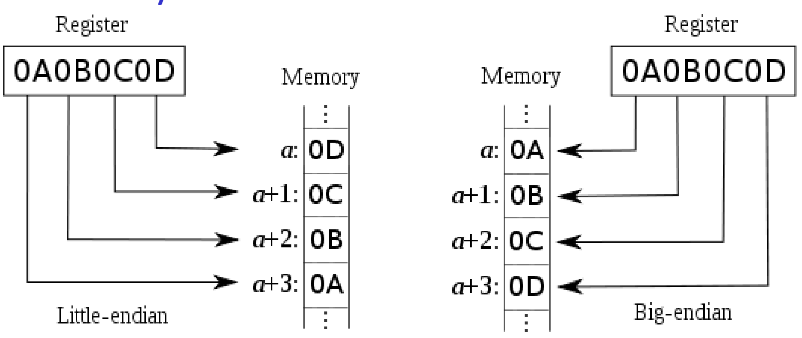
Memory Organization – cont'd

- Bytes are smallest addressable unit, but most data items use larger space: words
- In MIPS architecture a word is 32 bits or 4 bytes
  - 2^{32} bytes with byte addresses from 0 to 2^{32}-1
  - 2^{30} words with byte addresses 0, 4, 8, ... 2^{32}-4
- *Can a word start at any arbitrary byte address?*
- *What order the bytes in a word are stored?*



Endianness

- When a data entity is bigger than one byte (e.g. a 32-bit integer), how it is stored in memory?



6

How Information is Represented in Memory?

- Remember a (digital) computer can only understand binary information (0 and 1)
- How instruction and data are represented in memory?
  - Numbers?
  - Characters?
  - Strings?
  - Multimedia (audio, video)?
  - *Instructions (of a program)?*

Number Definitions

- Bits: *B*inary dig*it*
  - Two states (0 or 1)
- Nibble = 4 bits (not used much anymore)
- Byte or Octet = 8 bits
- Word
  - processor dependent: 4, 8, 16, 32, 64
  - The “natural” word length for a processor is the size of its register

Numbers in High-Level Languages

- char = 8 bits,
  short = 16 bits,
  int = basic size of processor register: 8, 16, 32, 64 bits
  long = 32 bits
  double = 64 bits
- Largest *number of items* represented in binary word with "n" bits is "2^n"
  - 2^8 = 256
  - 2^16 = 65,536
  - 2^32= 4,294,967,296

NOTE: number of items, NOT largest number *(Why?)*

9

Number Base

Common bases for representing a numeric values in computers

Name	*Base*	*Digits*	*For*
Binary	2	0, 1	machine
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	human
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	human
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	human

Numeric values can be represented in any base!

10

Number Representation

- Positional Representation

  $n = \sum (d_i \times base\hat{\ }i)$

  where d is a single digit less than base and "i" runs from 0 to number of digits – 1
- for i = 3, the number $d_3d_2d_1d_0$ =

  $(d_3 \times base\hat{\ }3) + (d_2 \times base\hat{\ }2) + (d_1 \times base\hat{\ }1) + (d_0 \times base\hat{\ }0)$
- Example: given the base-10 number 563

  hundreds tens units
  (5 x 10^2) + (6 x 10^1) + (3 x 10^0)
  (5 x 100) + (6 x 10) + (3 x 1)
  500 + 60 + 3

Base 2 (Binary) Numbers

- Positional notation:
  1) Numbered 0,1,2,…,n from right to left
  2) Rightmost binary position is "b0" and is the "*least_significant_bit*" (LSb) in the number
  3) Leftmost binary position is "bn-1" and is the "*most_significant_bit*" (MSb) in the number
- Example:

  generic form of a 8-bit binary number: b7-b0 = (0|1)
  b7x(2^7)+ b6x(2^6)+ b5x(2^5)+ b4x(2^4)+ b3x(2^3)+ b2x(2^2)+ b1x(2^1)+ b0x(2^0)

  b7x128 + b6x64 + b5x32 + b4x16 + b3x8 + b2 x 4 + b1 x 2 + b0 x 1
  b7 b6 b5 b4 b3 b2 b1 b0
  MSb LSb

Various Number Representation

Decimal	Binary	Octal	Hexadecimal
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Base Conversion

Convert binary to decimal:

1) Positional formula: base = 2 , the number 1011

= (1x2^3)+ (0x2^2) + (1x2^1)+ (1x2^0) base 2 number
= (1x8)+ (0x4) + (1x2) + (1x1) w/base 10 arithmetic
= 8 + 0 + 2 + 1
= 11

2) Alternate method: The multiplications can be replaced by a conditional add then sum the positional value at each non zero bit position

bn	bn-1		b16	b15	...	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
-----	------		-----	------		-----	-----	-----	----	----	----	---	----	---	---	---	---
2^n	2^(n-1)	...	2^16	2^15	...	2^11	2^10	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
------	------		-----	------		-----	----	----	----	----	----	---	----	---	---	---	---
			65536	32768		2048	1024	512	256	128	64	32	16	8	4	2	1
										0	1	0	0	1	0	0	1
			for **01001001**							**0**	**+ 64 +**	**0 +**	**0 +**	**8**	**+ 0 +**	**0 +**	**1**

73

Base Conversion – cont'd

Using *Horner's* nested form:

- o Rearrange the form of the basic positional formula (power series)

dn x (B^n) + dn-1 x B^(n-1) + … + d2 x (B^2) + d1 x (B^1) + d0 x (B^0)

= (dn x (B^n) + dn-1 x B^(n-1)) + … + d2 x (B^2) + d1 x (B^1) + d0 x (B^0)

= ((dn x B + dn-1) x B^(n-1)) + … + d2 x (B^2) + d1 x (B^1) + d0 x (B^0)

= ((…(((((dn x B + dn-1) x B + dn-2) x B + ….. + d2) x B + d1) x 2 + d0)

where B is the number base (e.g. 2, 10, 16,…)

- o Method (recursive evaluation)

1) This implies repeated division by *base B*, the first *remainder* will be the least significant digit (LSD) of the number

2) Then repeat with the *quotient* until the entire number is converted (i.e. the quotient is 0)

Example: Decimal -> Binary

Number		quotient	remainder	binary position
90	90/2	45	0	1
	45/2	22	1	2
	22/2	11	0	4
	11/2	5	1	8
	5/2	2	1	16
	2/2	1	0	32
	1/2	0	1	64

result = 1011010

(64+0+16+8+0+2+0 = 90)

Hex <-> Decimal

Hex -> decimal: same as binary to decimal but using 16 instead of 2

Positional formula: base = 16 , the number 138D

= (1x16^3) + (3x16^2) + (8x16^1) + (13x16^0)

= (1x4096) + (3x256) + (8x16) + (13x1)

= 4096 + 768 + 128 + 13

= 5005

Decimal -> hex:

Number		quotient	remainder
5005	5005/16	312	D (13)
	312/16	19	8
	19/16	1	3
	1/16	0	1

result = 138D

(1x4096) + (3*256) + (8*16) + (13*1) = 5005

How about Characters?

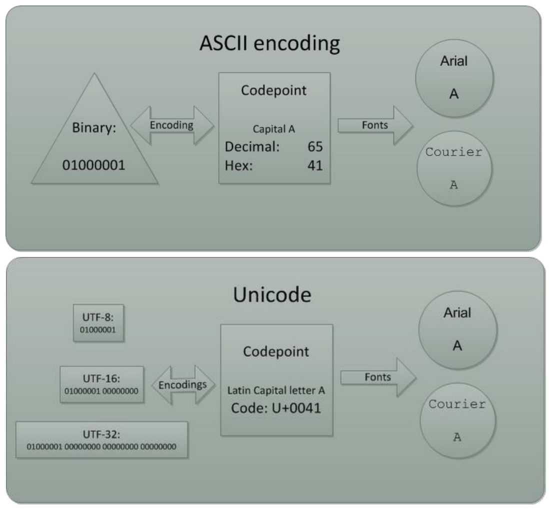
	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L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

The ASCII character set:

- Initially contained 128 7-bit encoded characters
- Including alphabetic, numeric, graphic and control characters
- Has been extended to include system or country specific characters (Unicode™ standard)
- *What character set do IBM computers use?*

How to handle Chinese characters?

Character Encoding: ASCII -> Unicode



UTF?

19

http://www.dotnetnoob.com

But I want to Store Strings!

- Strings can be represented as one-dimension arrays of characters
- How to signal the end of a string?
  - Length-prefix (e.g. in Pascal) – *what can be a problem?*
  - NULL (‘0’) terminated (e.g. in C language)
  - An explicit length field in the string object (e.g. C++)

length	F	R	A	N	K	k	e	f	w
5_{dec}	46_h	52_h	41_h	$4E_h$	$4B_h$	$6B_h$	65_h	66_h	77_h

F	R	A	N	K	NUL	k	e	f	w
46_h	52_h	41_h	$4E_h$	$4B_h$	00_h	$6B_h$	65_h	66_h	77_h

20

... and Multimedia (Audio, Video)

- Multimedia data are encoded into streams of bytes by encoders and stored in memory as sequences of bytes before they can be processed by computers!
- For example
  - MPEG2, MP4 for video
  - MP3 for audio
  - File Format (FF) standards in MPEG

21