B38DF
Computer Architecture and Embedded Systems

Alexander Belyaev

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering

E-mail: a.belyaev@hw.ac.uk
Office: EM2.29

Based on the slides prepared by Dr. Mustafa Suphi Erden

Measuring performance: Amdahl's law

1. A computer spend 30% of its time accessing memory, 20% performing multiplications, and 50% executed other instructions. As a computer architect, you have to choose between improving either the memory, multiplication hardware, or execution of nonmultiplication instructions. There is only one space on the chip for one improvement, and each of the improvements will improve its associated part of the computation by a factor of 2.

What speedup would making each of the three changes give?

Measuring performance: Amdahl's law

1. A computer spend 30% of its time accessing memory, 20% performing multiplications, and 50% executed other instructions. As a computer architect, you have to choose between improving either the memory, multiplication hardware, or execution of nonmultiplication instructions. Ther is only one spece on the chip for one improvement, and each of the improvements will improve its associated part of the computation by a factor of 2.

What speedup would making each of the three changes give?

$$T_{\text{new}} = T_{\text{old}}\left(1 - p + p/s\right) \quad S = T_{\text{old}}/T_{\text{new}} = \frac{1}{1 - p + p/s}$$

$$S_1 = \frac{1}{1 - 0.3 + 0.3/2} \approx 1.18 \quad S_2 = \frac{1}{1 - 0.2 + 0.2/2} \approx 1.11$$

$$S_3 = \frac{1}{1 - 0.5 + 0.5/2} \approx 1.33$$

Measuring performance: Amdahl's law

2. We are given a serial task which is split into four consecutive parts, whose percentages of execution time are $p1 = 0.1$, $p2 = 0.2$, $p3 = 0.3$, $p4 = 0.4$
We are told that:

1st part is sped up 2 times, so $s1 = 2$,
2nd part is sped up 5 times, so $s2 = 5$,
3rd part is sped up 20 times, so $s3 = 20$,
4th part is sped up 1.5 times, so $s4 = 1.5$.

Find the overall speedup.

Measuring performance: Amdahl's law

2. We are given a serial task which is split into four consecutive parts, whose percentages of execution time are $p1 = 0.1$, $p2 = 0.2$, $p3 = 0.3$, $p4 = 0.4$
We are told that:

1st part is sped up 2 times, so $s1 = 2$,
2nd part is sped up 5 times, so $s2 = 5$,
3rd part is sped up 20 times, so $s3 = 20$,
4th part is sped up 1.5 times, so $s4 = 1.5$.

Find the overall speedup.

$$T_{\text{old}} = T_1 + T_2 + T_3 + T_4 = p_1 T + p_2 T + p_3 T + p_4 T$$

$$T_{\text{new}} = \frac{p_1}{s_1} T + \frac{p_2}{s_2} T + \frac{p_3}{s_3} T + \frac{p_4}{s_4} T$$

$$S = T_{\text{old}} / T_{\text{new}} = 1 / \left(p_1 / s_1 + p_2 / s_2 + p_3 / s_3 + p_4 / s_4 \right) = 2.7$$

Error detection and correction

3. Assume we wish to create a code using 3 information bits, 1 parity bit (appended to the end of the information), and even parity. List all legal code words in this code. What is the Hamming distance of your code?

Error detection and correction

3. Assume we wish to create a code using 3 information bits, 1 parity bit (appended to the end of the information), and even parity. List all legal code words in this code. What is the Hamming distance of your code?

000 0

001 1

010 1

011 0

100 1

101 0

110 0

111 1

the Hamming distance = 2

Error detection and correction

4. Suppose we want an error-correcting code that will allow all single-bit errors to be corrected for memory words of length 10.

a) How many parity bits are necessary?

b) Assuming we are using the Hamming error-correcting code, find the code word to represent the 10-bit information word: 1001100110.

Error detection and correction

4. Suppose we want an error-correcting code that will allow all single-bit errors to be corrected for memory words of length 10.

a) How many parity bits are necessary?

b) Assuming we are using the Hamming error-correcting code, find the code word to represent the 10-bit information word: 1001100110.

*	*	1	*	0	0	1	*	1	0	0	1	1	0
1	***2***	3	***4***	5	6	7	***8***	9	10	11	12	13	14

0	0	1	1	0	0	1	1	1	0	0	1	1	0
1	***2***	3	***4***	5	6	7	***8***	9	10	11	12	13	14

Bit **1** checks bits 1, 3, 5, 7, 9, 11, 13

Bit **2** checks bits 2, 3, 6, 7, 10, 11, 14

Bit **4** checks bits 4, 5, 6, 7, 12, 13, 14

Bit **8** checks bits 8, 9, 10, 11, 12, 13, 14

Error detection and correction

5. We received the following string of bits which was originally generated by the Hamming algorithm: 1 0 1 0 1 0 1 1 1 1 0. It may contain a single bit error. Please check for an error and, if it is there, fix the error.

Error detection and correction

5. We received the following string of bits which was originally generated by the Hamming algorithm: 1 0 1 0 1 0 1 1 1 1 0. It may contain a single bit error. Please check for an error and, if it is there, fix the error.

1	0	1	0	1	0	1	1	1	1	0
1	***2***	3	***4***	5	6	7	***8***	9	10	11

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7

Bit **8** checks bits 8, 9, 10, 11

So we have the error in position 1+2+8=11.

The correct message is 1 0 1 0 1 0 1 1 1 1 1

Error detection and correction

6. We received the following string of bits which was originally generated by the Hamming algorithm: 1 0 1 0 1 0 1 1 1 0 1. It may contain a single bit error. Please check for an error and, if it is there, fix the error.

Error detection and correction

6. We received the following string of bits which was originally generated by the Hamming algorithm: 1 0 1 0 1 0 1 1 1 0 1. It may contain a single bit error. Please check for an error and, if it is there, fix the error.

1	0	1	0	1	0	1	1	1	0	1
1	***2***	3	***4***	5	6	7	***8***	9	10	11

So we have the error in position 2+8=10.

The correct message is 1 0 1 0 1 0 1 1 1 1 1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

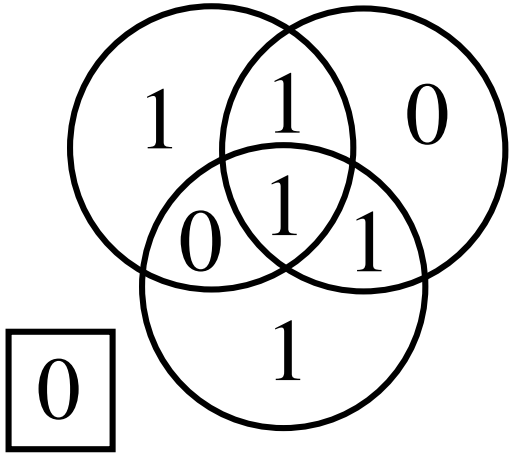
Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7

Bit **8** checks bits 8, 9, 10, 11

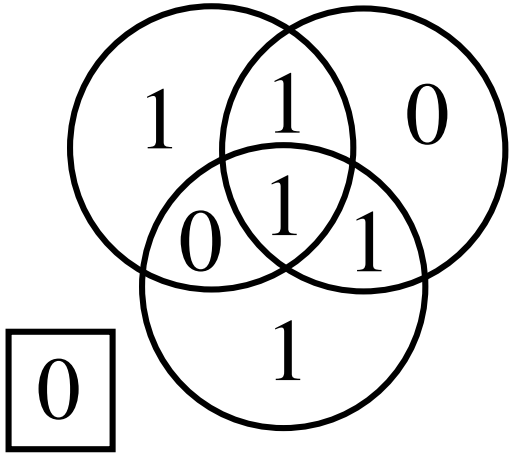
Error detection and correction

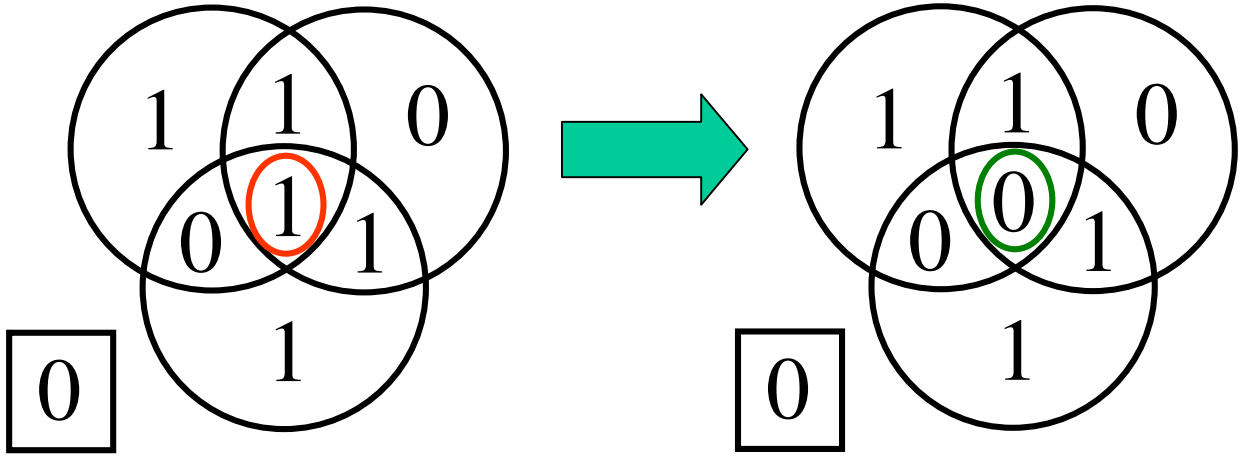
7. Check if there is a single/double error. In the single error case, correct the error.



Error detection and correction

7. Check if there is a single/double error. In the single error case, correct the error.





Memory

8. How many memory addresses can be referenced by an 16-bit microprocessor with 20 address lines?

a) 1 G
b) 1 M
c) 66 M
d) 32 M

Memory

8. How many memory addresses can be referenced by an 16-bit microprocessor with 20 address lines?

a) 1 G
b) 1 M
c) 66 M
d) 32 M

Memory

9. Assume you have a byte-addressable machine that uses 32-bit integers and you are storing the hex value 3456 at address 0.
a) Show how this is stored on a big endian machine.
b) Show how this is stored on a little endian machine.
c) If you wanted to increase the hex value to 123456, which byte assignment would be more efficient, big or little endian? Explain your answer.

Address →	00	01	10	11
Big Endian				
Little Endian				

Memory

9. Assume you have a byte-addressable machine that uses 32-bit integers and you are storing the hex value 3456 at address 0.
a) Show how this is stored on a big endian machine.
b) Show how this is stored on a little endian machine.

Address ⟶	**00**	**01**	**10**	**11**
Big Endian	00	00	34	56
Little Endian	56	34	00	00

Memory

10. Show how the following values would be stored by byte-addressable machines with 32-bit words, using little endian and then big endian format. Assume each value starts at address 0x10. Draw a diagram of memory for each, placing the appropriate values in the correct (and labelled) memory locations.

(a) 0x456789A1
(b) 0x0000058A
(c) 0x14148888

Address →	0x10	0x11	0x12	0x13
Big Endian				
Little Endian				

Memory

10. Show how the following values would be stored by byte-addressable machines with 32-bit words, using little endian and then big endian format. Assume each value starts at address 0x10. Draw a diagram of memory for each, placing the appropriate values in the correct (and labelled) memory locations.

(a) 0x456789A1
(b) 0x0000058A
(c) 0x14148888

Address ⟶	**0x10**	**0x11**	**0x12**	**0x13**
Big Endian	45	67	89	A1
Little Endian	A1	89	67	45

Address ⟶	**0x10**	**0x11**	**0x12**	**0x13**
Big Endian	00	00	05	8A
Little Endian	8A	05	00	00

Address ⟶	**0x10**	**0x11**	**0x12**	**0x13**
Big Endian	14	14	88	88
Little Endian	88	88	14	14

Compression

11. (a) Name an advantage of Huffman coding over LZ (Lempel-Ziv).
(b) Name an advantage of LZ over Huffman coding.
(c) Which is better?

(a) Huffman coding will usually result in better compression than LZ because it doesn't matter where in the message the characters are located.

(b) LZ is faster and can be done in hardware.

(c) It depends on what kinds of data you are dealing with and how you define "better." If "better" means "faster," LZ wins. If "better" means higher compression factor, Huffman will usually win.