Homework Chapter 5

John A. Gangemi
U6871-4612
CDA 3103

① Assume a byte-addressable machine that uses 32 bit integers and you are storing the hex value 1234 at address 0 :

A) Show how this is stored on a big endian machine

→ Address is 4 bytes long (32 bits = 4 bytes)

→ 1234_{16} to binary is $0001\ 0010\ 0011\ 0100_2$ (2 bytes)

		0011 0100	0001 0010	
00	00	34	12	
11	10	01	00	← Address

* Most significant byte is stored at the lowest address : $0001\ 0010_2$

B) Show how this is stored on a little endian machine

→ Address is 4 bytes long

→ 1234_{16} to binary is $0001\ 0010\ 0011\ 0100_2$ (2 bytes)

		0001 0010	0011 0100	
00	00	12	34	
11	10	01	00	← Address

* Least significant byte is stored at the lowest address :

$0011\ 0100_2$

C) If you wanted to increase the hex value to 123456, which byte assignment would be more efficient ?

→ Since the hex value is 3 bytes and address is 4 bytes then you must Little Endian b/c it allows for odd address reads & writes

② Byte-addressable machines with 32-bit words. Draw diagram of memory for Little Endian and Big Endian. Assume address starts at 10_{16}.

A) $4567\,89\,A1_{16}$ ⇒ binary is $\underbrace{0100\ 0101}_{\text{Byte 3}}\ \underbrace{0110\ 0111}_{\text{Byte 2}}\ \underbrace{1000\ 1001}_{\text{Byte 1}}\ \underbrace{1010\ 0001}_{\text{Byte 0}}{}_2$

Address →	0001 0000	0001 0001	0001 0010	0001 0011
Big Endian	45	67	89	A1
Little Endian	A1	89	67	45
	10_{16}	11_{16}	12_{16}	13_{16}

B) $0000058A_{16}$

Address →	0001 0000	0001 0001	0001 0010	0001 0011
Big Endian	00	00	05	8A
Little Endian	8A	05	00	00
	10_{16}	11_{16}	12_{16}	13_{16}

C) 14148888_{16}

Address →	0001 0000	0001 0001	0001 0010	0001 0011
Big Endian	14	14	88	88
Little Endian	88	88	14	14
	10_{16}	11_{16}	12_{16}	13_{16}

④ First two bytes of a 2M x 16 main memory have the following hex values:

Byte 0 is FE & Byte 1 is 01

If these bytes hold a 16 bit two's complement integer, what is its actual decimal value if:

A) memory is big endian?

→ Assuming Byte 0 is followed by Byte 1 ($FE01_{16}$)

→ Most significant byte is lowest address:

Address	00_2	01_2
	FE	01

$\Rightarrow$ $1111\ 1110\ 0000\ 0001_2$

$= -511_{10}$

B) memory is little endian?

→ Assuming Byte 0 is followed by Byte 1 ($FE01_{16}$)

→ Least significant byte is lowest address:

Address	00_2	01_2
	01	FE

$\Rightarrow$ $0000\ 0001\ 1111\ 1110_2$

$= 510_{10}$

⑤ What kinds of problems do you think endian-ness can cause if you wished to transfer data from a big endian machine to a little endian machine?

→ Given the definition of "endian", the byte order of multiple bytes for a single data element on a specific architecture, it is clear that conflicts between the two systems will arise as the byte-order interpretations are particular to their architecture. Also, as shown in the previous exercise (4) the decimal valve encodings are not an exact one-to-one correspondance and this could cause severe problems with storing and reading data.

⑧ A computer has 32 bit instructions and 12 bit addresses. Suppose there are 250 2-address instructions. How many 1-address instructions can be formulated?

→ Find the total # of bit patterns allowed for 32 bit instructions...

2^{32} instructions (bit patterns)

→ Find the # of bit patterns used thus far ...

250 2-address instrs $\Rightarrow$ $250 \times 2^{12} \times 2^{12} = (250)2^{24}$ bit patterns used

→ Number of remaining bit patterns ...

Total bit patterns - bit patterns used $= 2^{32} - (250)2^{24}$ bit patterns available

→ Let x = # of instrs with 1 address then ...

x 1-address instrs $\Rightarrow$ $x \cdot 2^{12} = (x)2^{12}$ bit patterns desired

$\Rightarrow$ bit patterns available = bit patterns desired

$$2^{32} - (250)2^{24} = (x)\,2^{12}$$

$$x = \frac{2^{32} - (250)2^{24}}{2^{12}} = 2^{20} - (250)2^{12}$$

$$= 2^{12}(2^{8} - 250) = \boxed{6 \times 2^{12}}$$

* There can be 6×2^{12} 1-address instructions

⑨ Convert the following expressions from infix to reverse Polish (postfix) notation.

A) $(8-6)/2 \Rightarrow [86-]/2 \Rightarrow \boxed{86-2/}$

B) $(2+3) \times 8/10 \Rightarrow [23+] \times 8/10 \Rightarrow [23+] \times [810/] \Rightarrow \boxed{23+810/\times}$

C) $(5 \times (4+3) \times 2 - 6)$

$\Rightarrow 5 \times [43+] \times 2 - 6 \Rightarrow [543+\times] \times 2 - 6 \Rightarrow \boxed{543+\times 2 \times 6 -}$

(17) Given instruction LOAD 1000. Assuming R1 is implied in the indexed addressing mode, determine the actual value loaded into the AC.

1000	1400
1100	400
1200	1000
1300	1100
1400	1300

R1

200

Mode	Value into AC	
Immeadiate	1000	(1000)
Direct	1400	(1000 = 1400)
Indirect	1300	(1000 = 1400 = 1300)
Indexed	1000	(1000 + 200)

(19) Non-pipelined system takes 200 ns to process a task. Same task can be processed in a 5-stage pipeline with a clock cycle of 40ns. Determine the speedup ratio of the pipeline for 200 tasks. What is the maximum speedup that could be achieved with the pipeline unit over the nonpipelined unit?

$n = 200, \quad K = 5, \quad t_p = 40\text{ns}, \quad t_n = K \cdot t_p$

⇒ The speedup ratio for 200 tasks:

$$\text{Speedup} = \frac{n t_n}{(K+n-1)t_p} = \frac{200(5 \times 40)}{(204)40} = \boxed{4.9}$$

⇒ The maximum speedup is the theoretical speedup and that is equal to the # of stages for a pipeline $(K) = \boxed{5}$

(20) A Non-pipelined system takes 100 ns to process a task. The same task can be processed in a 5-Stage pipeline with a clock cycle of 20 ns. Determine speedup ratio of the pipeline for 100 tasks. What is the theoretical speedup of the pipelined system over the non-pipelined system?

$n = 100,\ K = 5,\ t_p = 20ns,\ t_n = K \cdot t_p$

$\Rightarrow$ Speedup ratio for 100 tasks :

$$\text{Speedup} = \frac{n t_n}{(k+n-1)t_p} = \frac{(100)(5 \cdot 20)}{(104)\,20} = \boxed{4.8}$$

$\Rightarrow$ Theoretical speedup is equal to the # of pipeline stages $(k) = \boxed{5}$

(23) A digital computer has a memory unit with 24 bits per word. The ISA consists of 150 different operations. All instrs have an opcode and an operand (allowing for only 1 address). Each instr is stored in one word of memory.

→ Single instr = 24 bits thus 2^{24} total instrs

opcode	operand

} instr = 24 bits

A) How many bits are needed for the opcode?

→ Given 150 different operations, need n from $2^n \geq 150$

n = 8 bits for opcode

B) How many bits left for the operand?

→ 8 bits for opcode leaves (24 - 8) 16 bits for address

c) What is the maximum allowable size for memory?

→ word size = 24 bits = width

→ length = 2^{16} addresses = $2^6 \cdot 2^{10}$ = 64K

$\Rightarrow$ 64K x 24

D) what is the largest unsigned binary number that can be accommodated in one word of memory?

→ 1 word = 24 bits then $2^{24}-1$ is largest unsigned binary number

(24) The memory unit of a computer has 256K words of 32 bits each. The computer has an ISA format with 4 fields: opcode, mode, register address, memory address. Mode field specifies 1 to 7 addressing modes & register address specifies 1 to 60 registers. Assume an instr is 32 bits long.

opcode	mode	Register	memory	} instr = 32 bits
↑ (32-27) = 5 bits	↑ 3 bits	↑ 6 bits	↑ 256K = 18 bits	

A) Mode field = 3 bits B) Register field = 6 bits C) Address field = 18 bits

D) Opcode field = 5 bits

(25) An instruction takes 4 cycles to execute in a non-pipelined CPU: one cycle to fetch an instr, one cycle to decode instr, one cycle to perform ALU operation, & one Cycle to store the result. In a CPU with 4-stage pipeline, the instr still takes 4 cycles to execute, how can we say the pipeline speeds up the execution of the program?

→ Given a single instruction the non-pipelined system vs. the pipelined System is the same because for an instruction, non-pipelined is $nt_n = 1(K \cdot tp)$ and pipelined is $(K \cdot tp)$, and therefore both are equal. However if given a group of instructions the parallel execution of a pipelined system allows for a theoretical speedup of 4 times.

$\boxed{2}$

② The advantage of zero-address instruction computers is that they have short programs; the disadvantage is that the instructions require many bits, making them very long.

$\Rightarrow$ False

→ Zero-address instructions have only a single field for their ISA, the opcode. Given just an opcode field (fewer bits) the architecture must implement the use of a stack to perform operations that require two or three operands. Also, due to the nature of the zero address instruction of the stack architecture, as the # of operands decrease the # of instructions to execute code increases, thus the size for programs is larger than other operand storage architectures.

③ An instruction takes less time to execute on a processor using an instruction pipeline than on a processor without an instruction pipeline.

$\Rightarrow$ False

→ For single instruction programs the clock cycle time of a non-pipelined vs. pipelined system is the same. As shown by the equations for a single instruction, non-pipelined: $nt_n = n(k \cdot t_p)$ & pipelined: $k \cdot t_p$, with $n=1$ then the equations are equal.

$\boxed{3}$ Express answer in powers of two:

A) How many bytes are in 16TB?

$\Rightarrow$ $16 \cdot TB = 2^4 \cdot 2^{40} = 2^{44}$ bytes

B) How many bytes are in 20MB?

$\Rightarrow$ $20 \cdot MB = 5(2^2) \cdot 2^{20} = (5)2^{22}$ bytes

c) How many bytes are in 16GB - 40MB ?

$\Rightarrow$ 16 GB = $2^4 \cdot 2^{30} = 2^{34}$ bytes & 40MB = $(5)2^3 \cdot 2^{20} = (5)2^{23}$ bytes

$(2^{34} - (5)2^{23})$ bytes

D) How many nanoseconds are in 2^{20} microseconds ?

→ micro is 10^{-6} & nano is 10^{-9} thus the difference is 10^3.

$\Rightarrow$ $1048576 \times 10^3 = 1.048576 \times 10^9$ nanoseconds

$\boxed{4}$ How many unique values can be represented by :

A) A 2-bit value ? $\Rightarrow$ $2^2 = 4$ unique values

B) A 4-bit value ? $\Rightarrow$ $2^4 = 16$ unique values

c) An n-bit value ? $\Rightarrow$ 2^n unique values

$\boxed{5}$ What is the range of values that can be represented by an unsigned :

A) A 2-bit value ? $\Rightarrow$ 0 to 2^K-1 $\Rightarrow$ 0 to 2^2-1 = 0 to 3 values

B) A 4-bit value ? $\Rightarrow$ 0 to 2^K-1 $\Rightarrow$ 0 to 2^4-1 = 0 to 15 values

c) An n-bit value ? $\Rightarrow$ 0 to 2^n-1 values

$\boxed{6}$ If caching introduces latency on a cache hit, why don't computer designers simply eliminate caching & directly access memory on a memory access ?

→ Given a "Hit time", that is the time required to access the requested data in a given level of the memory hierarchy, the access time to main memory is still far greater than the access time for cache as shown in the memory hierarchy.

$\boxed{7}$ Example 6.2 ; Assume a byte-addressable memory consists of 2^{16} bytes with a cache of 16 blocks, where each block has 8 bytes.

→ Using Direct Mapped cache : [cache block Y ≡ main mem block X mod N]

→ Need # of blocks for main memory : 2^{16} bytes of memory / 8 bytes a block

$= 2^{16}/2^{3} = 2^{13}$ blocks of main memory

→ Bits for Main memory address is given by # of addressable bytes :

2^{16} bytes ⇒ 16 bits for address

→ Main memory format :

← 16 bits →

Tag	Block	Offset
9 bits	4 bits	3 bits

↓ # of remaining bits ⇒ 16-4-3 = $\boxed{9 \text{ bits}}$

↓ # of blocks in cache ⇒ 16 blocks = 2^4 blocks = 4 bits

↳ # of bytes/words in a block ⇒ 8 bytes = 2^3 bytes = 3 bits