Computer Systems Organization  
CSCI 463-1

Spring 2023

Homework #7

# I. Addressing

Assume byte-addressable memory in every case. Read the questions carefully – some are stated in bytes and some in bits. Except for small integers, give your answers as powers of 2.

You can use the examples on 4a2-12b through 4a2-13 as a guide:

1) If the problem is stated in bits, then convert it to bytes first. For example, slide 4a2‑12b shows:  
 4M of 32-bit words => 4M of 4-byte words

2) Then replace each abbreviation by the appropriate power of 2:  
 4M of 4-byte words = 4 \* 220 4-byte words = 4 \* 220 \* 4 bytes = 224 bytes

Here is an alternate solution to the same problem:  
 4M of 4-byte words = 4M \* 4 bytes = 16M bytes = 16MB = 16 \* 220 bytes = 224 bytes

Remember that you will have to memorize the prefixes used in the slides, so don’t leave it for the day before the exam. Do it now!

1a. If a memory chip has 64M of 4-byte words, how many bytes does it contain?

1b. How many address bits are required to address this memory?

1c. If a memory chip has 64M of 32-bit words, how many bytes does it contain?

1d. Is the answer to 1c the same or different from 1a? Why?

1e. If a memory chip has 64M of 64-bit words, how many bytes does it contain?

2a. If a memory chip has 16M of 8-byte words, how many bytes does it contain?

2b. How many address bits are required to address this memory?

2c. If a memory chip has 16M of 64-bit words, how many bytes does it contain?

2d. If a memory chip has 16M of 32-bit words, how many bytes does it contain?

3a. How many address bits do you need to address 64GB of memory?

3b. If a machine has 8 registers, how many bits are needed to address a register?

3c. If a machine has 64 instructions, how many bits are needed in an opcode (instruction code)?

# II. Cycle time and clock speed

Given the clock speed below, give the cycle time. Give your answer as an integer between 1 and 1000 followed by one of the standard abbreviations from the notes (e.g., ms, μs, ns, ps, etc.). You can use the examples on 4a2-9d and 4a2-9e as a guide.

4a. clock speed is 5 GHz

4b. clock speed is 100 MHz

4c. clock speed is 250 GHz

4d. clock speed is 500 MHz

4e. clock speed is 500 GHz

For each of these machines, given the clock cycle time, give the speed in Hz. Represent the speed as a number between 1 and 1000 followed by one of the standard abbreviations from the notes (e.g., Hz, GHz, MHz, etc).

5a. clock cycle time is 1.25ns

5b. clock cycle time is 1.5ns

5c. clock cycle time is 1μs

5d. clock cycle time is 800ms

5e. clock cycle time is 25ps

# III. Pipelining

The slides contain five examples. Study them first:

Example 1 (no pipeline): slides 5d-31b and 31c.

Example 2 (full pipeline): also on slides 5d-31b and 31c.

Example 3 (different-length steps): slide 31e.

Example 4 (second copy of one step, and it doubles the speed): slide 31f.

Example 5 (second copy of one step, but it doesn’t double the speed): slide 31g.

Then study the restaurant analogy provided on Blackboard.

6. Answer the following questions:

a) In example 5 (slide 5d-31g), why does duplicating the execution hardware not double the speed of the system?

b) Is it possible that duplicating the execution hardware wouldn’t change the speed of the system at all? Why or why not?

c) Is is possible that duplicating the execution hardware would make the system slower? Why or why not?

Finally, work the following problems.

This table gives three instruction types with the times for each step of a five‑step fetch/execute pipeline.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Fetch opcode | Decode | Fetch operands | Execute | Writeback | CPU  speed |
| 7. | 1 cycle | 1 cycle | 1 cycle | 1 cycle | 1 cycle | 1 Ghz |
| 8. | 1 cycle | 1 cycle | 2 cycles | 8 cycles | 2 cycles | 10 Ghz |
| 9. | 1 cycle | 1 cycle | 3 cycles | 3 cycles | 0 cycles | 100 MHz |

For each instruction type, calculate the following. Use units as in previous assignments, i.e., represent time as an integer (rounded) between 1 and 999 multiplied by a standard abbreviation such as ms, μs, ns, or ps.

Represent instructions per second as an integer (rounded) between 1 and 999 multiplied by a power of 10 whose exponent is a multiple of 3, e.g., 103, 106, etc. If the integer is 1, it may be omitted, i.e., you can write 106 instead of 1 x 106.

Remember that instructions per second is the reciprocal of time per instruction, i.e., c), e) and g) are the reciprocals of b), d) and f) respectively. Also remember that cycle time is the reciprocal of the CPU speed.

a) What is the cycle time?

b) If pipelining is not supported, give the time it takes to process one instruction.

c) Calculate the number of instructions per second.

d) If pipelining is supported and the pipeline is kept full, how often is an instruction completed?

e) Calculate the number of instructions per second.

f) If there are two parallel copies of the execution step and the pipeline is kept full, how often is an instruction completed?

g) Calculate the number of instructions per second.

# IV. Stack machines

The slides discuss three types of machine architecture, accumulator, GPR and stack machines.

In an accumulator machine, the result of an operation is always in the accumulator. There can be at most one other operand, expressing the value that we are going to add to (or subtract from, etc.) the accumulator. MARIE is an accumulator machine.

The IBM mainframe is a GPR machine, so you have already studied those too. It is a memory-memory machine since it allows memory-memory instructions, namely the SS and SI instructions. If we eliminated those instructions, we would have a register-memory machine. If we also eliminated all the RX and RS instructions except for load and store instructions, we would have a load-store machine.

So the only type of architecture we have not studied is the stack architecture. In this problem we will write some code for an imaginary stack machine. Here are the instructions:

PUSH X push contents of memory address X on stack

STORE X store contents of top of stack in m(X) (does not pop stack)

POP pop stack

ADD add contents of top two stack cells, pop them, push result on stack

MULT multiply (details same as ADD)

SUB subtract (details same as ADD)

DIV divide (details same as ADD)

For all of the arithmetic operations, the operator on top of the stack (i.e., the one that was pushed second) is the second operand of the expression.

After you store the final result, make sure to pop the stack. (Otherwise the stack will continue to grow and grow as your program runs.)

Start by studing the example on slide 5c‑15. (You may find the slides before and after helpful also.)

10. Consider these two programs:

PUSH X PUSH Y

PUSH Y PUSH X

DIV DIV

STORE Z STORE W

POP POP

a) Write an equation for Z

b) Write an equation for W

c) Does Z = W? Why or why not?

11. Convert each of the following to postfix notation.

a) A + B + C + D i.e., ((A + B) + C) + D

b) A + (B + C + D) i.e., A + ((B + C) + D)

c) A + (B + (C + D))

12. Now write code for each of the expressions below. Remember that they are written in standard mathematical notation, i.e. multiplication and division precede addition and subtraction.

Note that you will have to convert the expressions to postfix (reverse Polish) notation first.

Do not reorder the calculations. If the problem asks for A + B, do not use the commutative law to replace that with B + A. Remember that A + B + C means (A + B) + C, not A + (B + C). There are two reasons you should evaluate the expressions as written. First, you are trying to emulate the compiler. Second, although these pairs of expressions are equal for the small integers we are practicing with, that is not necessarily true for complex floating point calculations.

a) X = A + B/C

b) Y = (A + B)/C

c) Do parts a) and b) calculate the same value, i.e., does X = Y? Why or why not?

d) X = (A + B) \* (C - D)

e) X = A \* (B + C / D)