# CSCI 5593 - Homework 1

04-FEB-2019

Ranjan Raut

1. Semester Individual Website Assignment:

→ Raut , Ranjan , ranjan.raut@ucdenver.edu , https://github.com/rautranjan/CSCI-5593_Advanced_Computer_Architecture

---

2. Consider a load-store type machine with the following specifications:
   - $2^{32}$ x bytes of memory
   - 32-bit fixed format instructions
   - 32 32-bit general purpose registers (GPR)
   - 3-address register-to-register arithmetic instructions
   - Single address mode for load/store: base + displacement
   - Capable of performing a total of 32 arithmetic operations

   For simplicity assume that the machine only performs arithmetic operations plus data transfer operations (i.e. load and store).

   A. Write the equivalent machine level language corresponding to a C statement of C = A + B

   →

   1. LOAD R1, A
   2. LOAD R2, B
   3. ADD R3, R1, R2
   4. STORE R3, C

   B. Give an instruction format for the arithmetic operations. To do this draw a diagram of the instruction format with each field clearly specified. For each field indicate its size, the reason for selected size, and a description of what purpose the field serves.

   →

   For arithmetic operations, we can have R-type (Regular) instruction format with following constructs:
   - Total instruction size will be 32 bits since its 32-bit fixed instruction format
   - Registers - We'll have 3 fields for registers (i.e. RS, RT, RD) since we'll be using 3 address instruction formats. Each of these register fields will be of size 5 bits, since we are having total 32 registers (i.e. $2^5$)
     - RS - Register Source: - First source register
     - RT - Register Target: - Second Source Register
     - RD - Register Destination: - Target Register where result will be stored
   - OPCODE - Since, we can have total 32 arithmetic operations (i.e. $2^5$), OPCODE field will require size of at least 5 bits + we'll reserve one more bit in OPCODE to specify address mode. Hence, OPCODE will require total 6 bits.

    - The opcode is the machine code representation of the instruction. Several related instructions can have the same opcode
- Now, we are still left with 11 bits in instruction. We'll assign 5 bits for SHAMT (Shift Amount) and 6 bits for FUNCT (Function)
    - SHAMT - Used with the shift and rotate instructions, this is the amount by which the source operand RS is rotated/shifted.
    - FUNCT - For instructions that share an opcode, the FUNCT parameter contains the necessary control codes to differentiate the different instructions. In our case, it'll be useless parameter

| OPCODE | RS | RT | RD | SHAMT | FUNCT |
|---|---|---|---|---|---|
| *6 bits* | *5 bits* | *5 bits* | *5 bits* | *5 bits* | *6 bits* |

**Instruction Format Diagram**

C. Give an instruction format for the load/store operations. To do this draw a diagram of the instruction format with each field clearly specified. For each field indicate its size, the reason for selected size, and a description of what purpose the field serves.

→

For Load/Store type of operations, we can have I-type (Immediate) type instruction format with following constructs:

- Total instruction size will be 32 bits since its 32-bit fixed instruction format
- Registers - We'll have 2 fields for registers (i.e. RS, RT) since we'll be using LOAD/STORE instruction type. Each of these register fields will be of size 5 bits, since we are having total 32 registers (i.e. $2^5$)
    - RS - Register Source
    - RT - Register Target
- OPCODE - Since, we can have total 32 arithmetic operations (i.e. $2^5$), OPCODE field will require size of at least 5 bits + we'll reserve one more bit in OPCODE to specify address mode, in our case base + displacement. Hence, OPCODE will require total 6 bits.
    - The opcode is the machine code representation of the instruction. Several related instructions can have the same opcode
- IMM – We'll assign remaining 16 bits for displacement.
    - IMM is the 16-bit immediate value. We add the value to the Base Register, hence called displacement.

| OPCODE | RS | RT | IMM |
|---|---|---|---|
| *6 bits* | *5 bits* | *5 bits* | *16 bits* |

**Instruction Format Diagram**

# CSCI 5593 - Homework 1

3. To see how different ISA decisions will impact the machine design, consider designing an accumulator machine with the following specifications:
   - $2^{24}$ words of memory [words are 32-bit wide]
   - Fixed format instructions
   - A 32-bit accumulator register (AC)
   - An index register X
   - Index address mode: address field + X when indexing is indicated in the instruction
   - Capable of performing a total of 128 operations

   A. Write the equivalent machine level language corresponding to a C statement of C = A + B

   →

   1. LOAD A
   2. ADD B
   3. STORE C

   B. Give an instruction format for this computer. To do this draw a diagram of the instruction format with each field clearly specified. For each field indicate its size, the reason for selected size, and a description of what purpose the field.

   →

   For operations in accumulator, we can have instruction format with following constructs:

   - OPCODE - Since, we can have total 128 operations (i.e. $2^7$), OPCODE field will require size of 7 bits.
     - The opcode is the machine code representation of the instruction.
   - Index (X) - 1 bit for index register (to indicate indexing in instruction). It can have either 0 or 1 value.
     - if value is 1, then value of register X will be considered as index address Hence, address = offset + value of X
   - IMM – IMM is immediate address (can be considered as Base address) and it will have 24 bits, each address representing location of word.

| OPCODE | INDEX *X* | IMM |
|---|---|---|
| 7 bits | 1 bit | 24 bits |

**Instruction Format Diagram**

# CSCI 5593 - Homework 1

4. When designing memory systems, it becomes useful to know the frequency of reads versus writes as well as the frequency of accesses for instructions versus data. Using the average instruction-mix information for MIPS for the program spice (as given below),

| Instruction Class | MIPS Examples | HLL Correspondence | Frequency (spice) |
|---|---|---|---|
| Arithmetic | add, sub, addi | operations in assignment statements | 50% |
| Data transfer | lw, sw | references to data structures | 41% |
| Conditional branch | beq, bne, slt, slti | If statements and loops | 8% |
| Jump | j, jr, jal | procedure calls/returns, case/switch statements | 1% |

find the following:

A. The percentage of all memory accesses that are for data (vs. instructions)

→

From the table above, we can see that lw (LOAD) and sw (STORE) both contribute to 41%. Therefore,

$$= \frac{41\%}{1 + 41\%} \times 100 = \frac{0.41}{1 + 0.41} \times 100 = \frac{0.41}{1.41} \times 100$$

$$= 0.2907 \times 100 = \underline{29.07\ \%}$$

| **The percentage of all memory accesses that are for data (vs. instructions)** | **29.07 %** |
|---|---|

B. The percentage of all memory accesses that are reads (vs. writes). Assume that two-thirds of data transfers are loads.

→

From the given data, lw (LOAD) and sw (STORE) both contribute to 41%. As per assumption, 2/3 of data transfers are loads (reads)
i.e. [ 41 X 2 % 3 ] = ***27.33 %*** read operation

The percentage of all memory accesses that are reads (vs. writes)

$$= \frac{1 + 27.33\%}{1 + 41\%} \times 100 = \frac{1 + 0.2733}{1 + 0.41} \times 100 = \frac{1.2733}{1.41} \times 100$$

$$= 0.903 \times 100 = \underline{90.3\ \%}$$

| **The percentage of all memory accesses that are reads (vs. writes)** | **90.3 %** |
|---|---|