# Project Human Sorter Handout

#### Hardware design

It is helpful to list the components of the system, as shown in Table 1, which shows that the system consists of 5 components. The controller component is played by a student, who in one step decodes the current instruction, controls other components to execute the current instruction, and determine the next instruction.

**Table 1** A human sorter computer consisting of seven components

|  |  |  |
| --- | --- | --- |
| **Component** | **Implemented by** | **Functionality of the component** |
| Controller | 1 student | Decode instruction, control components to execute instruction, and determine next instruction |
| Operation Unit and Memory in one unit | N students in N positions | Store data and actually execute instructions under the control of the controller |
| Overseer | 1 student | Ensure three correctness; stop the program when any violation occurs |
| … | … | … |
| … | … | … |
| … | … | … |
| … | … | … |

#### Instruction set design

An instruction set should consist of around a dozen instructions, usually 7~20. If your design needs 40 instructions, something is wrong. It helps to explicitly list the instruction set, alphabetically by opcode. Also note that you are designing a human computer, instead of an electronic computer.

**Table 2** Instruction set of the student’s human sorter computer, needing 9 instructions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Operand1** | **Operand2** | **Operand3** | **Explanatory remarks** |
| HALT | N/A | N/A | N/A | The program terminates |
| MOV | Immediate value | Register1 | N/A | Set Register1 to the Immediate value  E.g., MOV 0, R1 means 0🡪R1 |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |

#### The sort program

Each student must provide an assembly language program to accomplish the sort task, as demonstrated in Table 3. It happens that the program’s LoC (lines of code) is 23 (your program may have a different LoC count). This LoC is the *static* length of the program. When the program executes, it may execute more instruction cycles, such as 129 steps (instructions), which is the *dynamic* length of the program’s execution.

In addition, the student can provide supplementary information to highlight any distinct features of the design. For instance, my human sorter utilizes an XXX feature of the YYY instruction, which takes advantage of the ZZZ capability of humans.

**Table 3** Assembly language code of the sort program executed by the human sorter computer

|  |  |  |
| --- | --- | --- |
| **No.** | **Instruction** | **Explanatory Comments** |
| 1 | MOV 17, R1 | Initialize R1 to N, the number of data items to be sorted |
| 2 | … | … |
| 3 | … | … |
| 4 | … | … |
| 5 | … | … |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 |  |  |
| 21 |  |  |
| 22 |  |  |
| 23 |  |  |

#### Records of executions

Each student must provide meticulous records of at least two executions. Here, “meticulous” means that each step should be recorded. A sample record table is shown below. Information such as the following should be provided:

* Initial configuration of the computer. E.g., the program of the sequence of 23 instructions is stored in the controller’s brain; execution starts at instruction No. 1; students of the data group are sorted by name.
* Configuration of the human sorter computer after every step.
* Total number of steps, etc.

Recall that each execution must satisfy three criteria of correctness:

* Result correctness: the students are indeed ordered by height
* Algorithmic correctness: the execution implements the quicksort algorithm
* Systems correctness: the team computer executes the program sequentially, i.e., step-by-step, one instruction after another

**Table 3** Record of execution 1.   
Note that Record of execution 2 may show a different number of steps

|  |  |  |
| --- | --- | --- |
| **Step** | **Instruction Executed** | **Explanatory Comments** |
| 0 | Initial configuration | Students of the data group are sorted by name; N=16 |
| 1 | MOV 17, R1 | Initialize R1 to N, the number of data items to be sorted |
| 2 | … | … |
| … | … | … |
| … | … | Start of the first iteration of the main loop |
| … | … | …; comparison #1; … |
| … | … | … |
| … | … | End of the first iteration of the main loop |
| … | … | Start of the second iteration of the main loop |
| … | … | …; comparison #2; … |
| … | … | … |
| … | … | End of the second iteration of the main loop |
| … | … | … |
| … | … | … |
| … | … | Start of the last iteration of the main loop |
| … | … | … |
| … | … | … |
| … | … | End of the last iteration of the main loop |
| … | … | … |
| 129 | Halt | End of program; the data group are now sorted by height |