

Project: Report Document

Daniel Alejandro Marin - R11858881

Texas Tech University
CS3375 - Computer Architecture
Instructor: Dr. Juan Carlos Rojas

December 6th, 2024

Contents

1	Introduction	2
2	Design	2
2.1	Instruction Scheduler Abstract Class	2
3	Methodology	3
4	Tests	3
5	Results	3
6	Discussion	3

- `logger`: this property contains a ‘log book’ of the logs performed by the scheduler, used for debug statements and keeping track of instructions in the order they retire (outputs).

These are the properties all schedulers contained, additional properties can be added to the subclasses of `InstructionScheduler` when needed. Now, let’s look at the methods defined in this parent class.

- `__init__(self, functional_units=1)`: this constructor method is used to create instances of this class, and it is called by the children classes as to initialize the aforementioned properties.
- `add_instruction(self, instruction)`: this method is in charge of adding an instruction to instructions. It is used to add parsed instructions set that we want to simulate the scheduling.
- `execute_cycle(self)`: this method contains the logic of executing a cycle in the simulation incrementing `current_cycle` by one, attempting to schedule instructions, and retiring instructions. The definition is as follows:

```
def execute_cycle(self):
    self.current_cycle += 1
    self.schedule()
    self._retire_instructions()
```

- `run(self)`: this method is fairly straightforward, it executes cycles until all instructions have been dispatched out of the scheduler. Most configurations used the definition:

```
def run(self):
    while self.instructions or self.instructions_in_progress:
        self.execute_cycle()
```

- `schedule(self)`: this method contains the logic for scheduling instructions. It is abstract, since each subclass configuration schedules instructions differently.
- `_schedule_instruction(self, instr : Instruction)`: this method is in charge of updating the status of an instruction whenever it gets scheduled. It does so in the following manner:

```
def _schedule_instruction(self, instr : Instruction):
    instr.issue_cycle = self.current_cycle
    instr.exp_completion = self.current_cycle + instr.latency()
    instr.started = True
    self.instructions_in_progress.append(instr)
```

3 Methodology

4 Tests

5 Results

6 Discussion