Exercises for Architectures of Supercomputers

5th Exercise, 4./5.12.2019

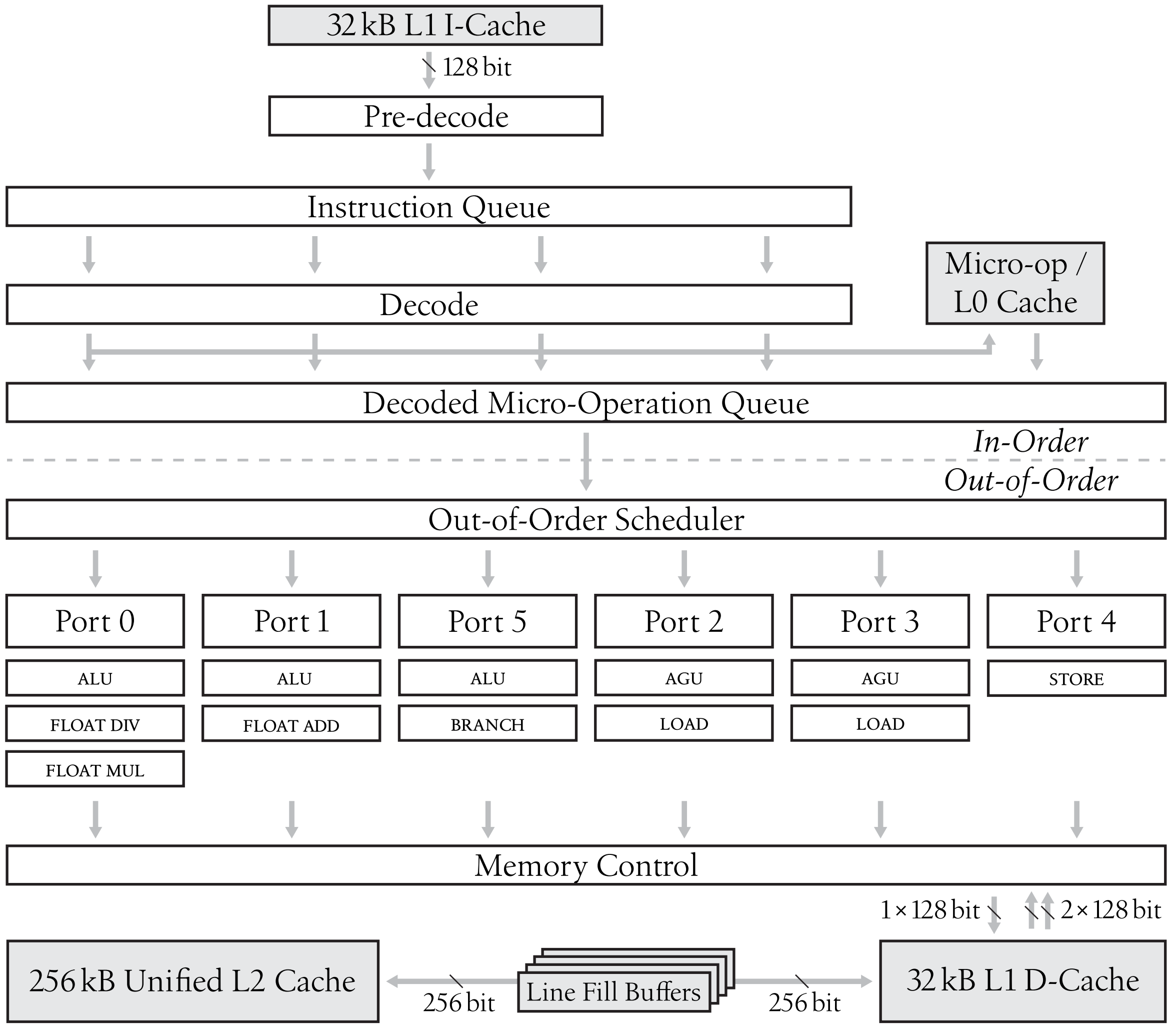
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Superscalar Cores

- To increase instruction throughput, superscalar cores contain multiple “functional units”
  - Functional units: general name for units implementing logic corresponding to some instruction
    - Load unit: load data from memory into registers
    - Store unit: store data from registers to memory
    - ALU (arithmetic-logical unit): Integer arithmetic, logical operations
    - FP ADD unit: floating-point addition unit
    - FP MUL unit: floating-point multiplication unit
    - …
- To keep multiple functional units busy, other/all stages of the instruction pipeline are duplicated
  - IF: Need to fetch more than one instruction per cycle
  - ID: Need to decode more than one instruction per cycle
  - …

Superscalar core of Intel Ivy Bridge-EP



Exercise

- This week, we'll try to determine the number of 'FP MUL' (floating-point multiplication) units of the
  - Intel Ivy Bridge-EP microarchitecture (used in Emmy's processors); and
  - Intel Broadwell-EP microarchitecture (used in Meggie's processors)

- To this end
  1. We first measure the instruction latency of the FP MUL instruction on both processors using a loop with no unrolling
  2. Next, we perform the necessary loop unrolling so each core executes **exactly one** FP MUL instruction per cycle
  3. Finally, we'll try doubling/tripling/… the unrolling factor to find out whether the core is capable of executing **multiple** FP MUL instructions per cycle

1. Measuring the FP MUL latency

- You can use the following function to measure the latency of the floating-point multiplication instruction

```
double benchmark(double *A, int N) {
  int i;
  double result = 1.0;
  #pragma novector
  #pragma nounroll
  for (i=0; i<N; ++i)
    result = result * A[i];
  return result;
}
```

- Hints
  - The array should contain around 3000 elements
  - Initialize all array elements with a value of 1.0 before calling `benchmark()`
  - Measure the runtime (in cycles per loop iterations) the same way as last week:
    $t_{cy/it} = t_{sec} \cdot f_{core} / n_{reps} \cdot 3000$

2. Getting to one FP MUL per cycle

- To get the processor to execute exactly one FP MUL instruction per cycle, adjust the code so the unrolling factor matches the instruction's latency
    - Example: Floating-point addition on Ivy Bridge (Emmy)
        - The FP ADD latency was three cycles
        - With three-way unrolling, the processor can execute one FP ADD per cycle
    - To avoid data hazards at runtime which lead to stalls in the pipeline, make sure to introduce distinct partial products for each multiplication
        - This way the compiler uses different registers for each instruction

3. More than one FP MUL per cycle?!

- Try doubling the unrolling factor
  - Does the instruction throughput (i.e., the number of FP MUL instructions per cycle) change on the Ivy Bridge processor?
  - Does the instruction throughput change on the Broadwell processor?
  - What do you make of your observations?

- Try tripling the unrolling factor
  - Does the instruction throughput change on any of the processors?
  - What do you make of your observations?

Hints: Accessing the Meggie cluster

- In this week's exercise, you're supposed to use nodes from the Emmy and Meggie cluster
- You already know how to request nodes from the Emmy cluster
- Use the following instructions to request a node from the **Meggie** cluster
  - Log into the head node for the Meggie cluster: `meggie.rrze.fau.de`
  - Instead of PBS, the Meggie cluster uses SLURM as batch management system, so instead of `qsub`, use the `srun` binary to request a node:

```
$ srun -p work --time=1:00:00 --ntasks-per-node=1
    --cpus-per-task=20 --nodes=1 --exclusive
    --constraint=hwperf --cpu-freq=2200000 --pty bash -l
```

- **Hint:** If you have trouble compiling your code on the Meggie cluster, compile it on the Emmy cluster. Both clusters share the same NFS home directories, so you don't have to copy any files around