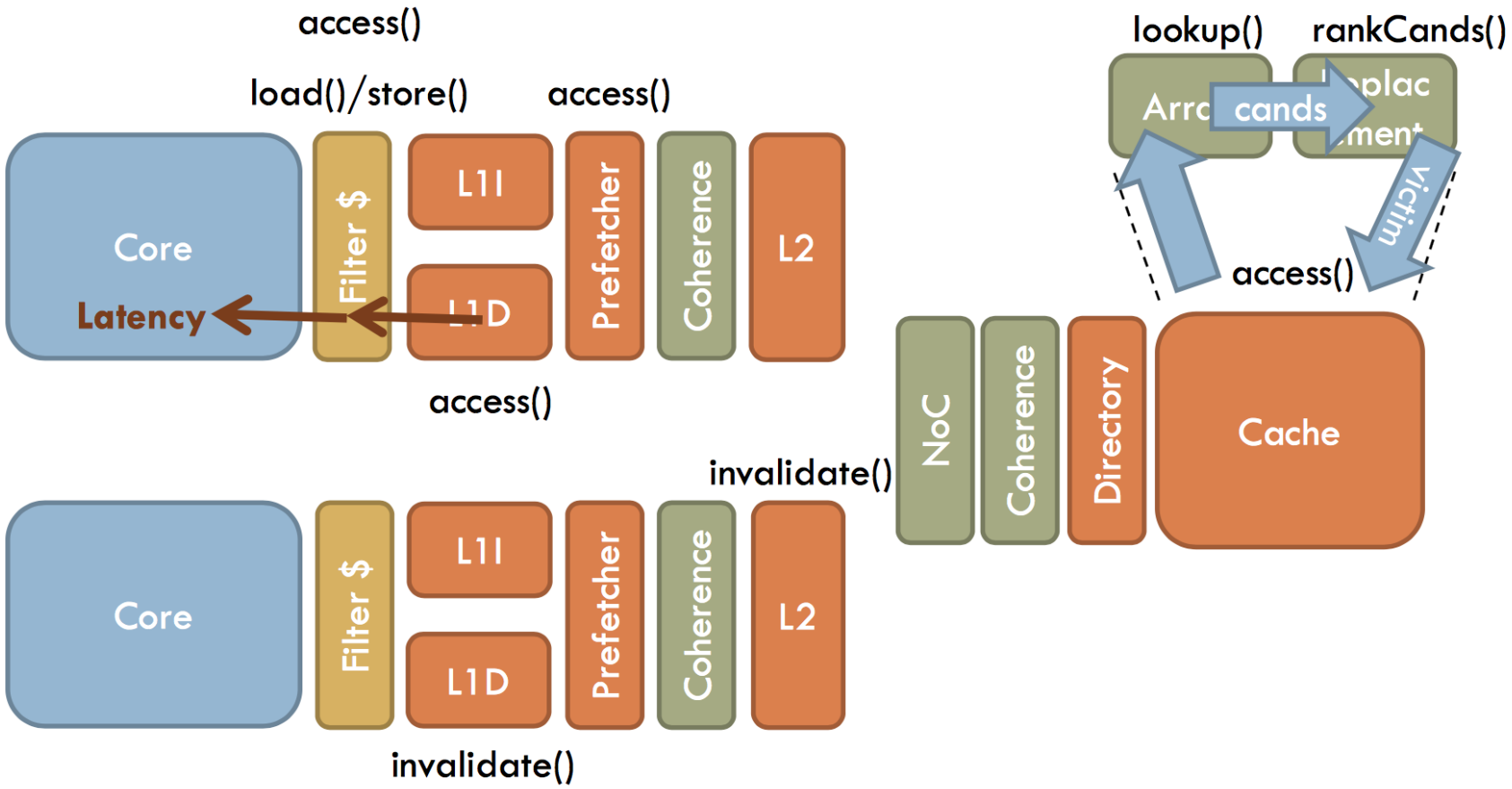
CSCE614 HW4:
Cache Access Modeling in ZSim

Cache Hierarchy in ZSim



Memory Acess in A Cache

- Access cache:
  - If HIT: coherence controller porcessAccess
  - If MISS:
    - Find a victim in a set (consult the replacement policy)
    - Coherence conroller processEviction
    - Replace with the new line

Important Classes

- Cache `(cache.h[.cpp])`
  - CacheArray, ReplPolicy, (CC but can ignore now)
  - access()
- CacheArray `(cache_array.h[.cpp])`
  - Address* (tag array), ReplPolicy
  - lookup(), preinsert(), postinsert()
- ReplPolicy `(repl_policies.h[.cpp])`
  - update(), replaced(), rankCands()

Cache::access()

- Access a cache to perform load or store operations

```
Cache::access() {
```

Look Up Cache (array->lookup)

Cache Miss Handling (including Block Replacement / Writeback)

Cache Hit Handling (update replacement)

```
}
```

`Cache::access()` (Cont'd)

- Access a cache to perform load or store operations

```
Cache::access()  {
```

```
Look Up Cache (array->lookup) {
    If find the line {
        Update repl_policy and return line ID;
    } else {
        return -1;
    }
}
```

```
    . . . . . .
}
```

Cache::access() (Cont'd)

- Access a cache to perform load or store operations

```
Cache::access() {
        Cache Miss Handling (including Block Replacement / Writeback)
         {
                array->preinsert(); // consult repl_policy to find a victim
                cc->processEviction(); // write back if needed
                array->postinsert(); // finish and update the replacement
         }
      ......
}
```

Read cache_array.h[.cpp](SetAssocArray) for lookup(), preinsert(), and postinsert();

SetAssocArray

lookup(), preinsert(), postinsert()

```
SetAssocArray::lookup() {
        if found: // hit
                repl_policy->update() and return lineID
        else: // miss
                return -1
}
```

```
SetAssocArray::preinsert() {
        repl_policy->rankCands() to find a victim
}
```

```
SetAssocArray::postinsert() {
        repl_policy->replaced(); // replace block
        update tag array
        repl_policy->update();   // update newly inserted block
}
```

ReplPolicy

update(), replaced(), rank()

ReplPolicy::update () {
    updated when a hit during CacheArray::lookup()
    or
    updated in miss handling after replacement is done (postinsert)
}

ReplPolicy::replaced() {
    replace the new line, may need to update some replacement info
}

ReplPolicy::rank() {
    find one victim from given candidates (find a victim from a set)
}

SRRIP implementation

- Add the member variables in `rrip_repl.h`
- Implement the member methods
  - `update()`
    - need to differentiate newly insert block or old block
  - `replaced()`
  - `rank()`
- Refer to `repl_policies.h` (LRU)

Initialize SRRIP

- Add constructor in `init.cpp`

```
      ......
} else if (replType == ”SRRIP") {
        // max value of RRPV, you need to pass it to your SRRIP constructor
        uint32_t rpvMax = config.get<bool>(prefix + "repl.rpvMax", 3);
        assert(isPow2(rpvMax + 1));
        // add your SRRIP construction code here

 } ......
```