به نام خداوند بخشنده مهربان

# معماری کامپیوتر

دانشکده مهندسی کامپیوتر

دکتر اسدی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶ و امیرحسین صوری، ۴۰۱۱۰۶۱۸۲

## تمرین دهم سوال اول عملی

۱. **تمرین عملی اول**

در این تمرین قصد داریم یک برنامه محک[۹] را با سیاست‌های مختلف را روی ابزار Gem5 شبیه‌سازی کنیم و نتایج بدست آمده از شبیه‌سازی را تحلیل کنیم.

برای شبیه‌سازی از دو سیاست FIFO و LRU استفاده کنید. برای پیکربندی سیستم شبیه‌سازی نیز از پیکربندی پیش‌فرض در آدرس زیر استفاده کنید و به صورت زیر با قابلیت‌های زیر اجرا کنید.

```
./build/X86/gem5.opt configs/deprecated/example/se.py -c [bench] --
    caches --l2cache --l2_size=4kB --mem-type=DDR4_2400_16x4 --
    cacheline_size 128
```

پس از شبیه‌سازی به واسطه راهنمایی داده شده، دنباله[۱۰] دسترسی حافظه توسط شبیه‌ساز را بدست آورید.

حال با توجه به این دنباله دسترسی به حافظه، به واسطه یک اسکریپت پایتون، بدست آورید که در صورتی که سیاست حافظه نهان به صورت بهینه عمل می‌کرد، مقدار نرخ برخورد به چه صورت تغییر می‌کرد.

**نکته:** فرض کنید ساختار حافظه نهان به صورت 2-way set-associative است.

**راهنمایی(اضافه کردن قابلیت سیاست حافظه نهان)**

برای اضافه کردن قابلیت سیاست جایگزینی[۱۱] هنگام اجرای شبیه‌سازی مراحل زیر را اجرا کنید. ابتدا در فایل موجود در مسیر configs/common/ObjectList خط زیر را اضافه کنید:

```
repl_list = ObjectList(getattr(m5.objects, 'BaseReplacementPolicy',
    None))
```

سپس قابلیت‌های زیر را به عنوان قابلیت‌های شبیه‌سازی، در فایل configs/common/Options.py به تابع addNoISAOptions اضافه کنید:

```
parser.add_argument("--l2_repl", default="LRURP",
choices=ObjectList.repl_list.get_names(),
help = "replacement policy for l2")
```

نهایتا سه خط زیر را به انتهای تابع _get_cache_opts در فایل موجود در مسیر configs/common/CacheConfig.py اضافه کنید:

```
replacement_policy_attr = f"{level}_repl"
if hasattr(options, replacement_policy_attr):
opts["replacement_policy"] = ObjectList.repl_list.get(getattr(options,
    replacement_policy_attr))()
```

حال می‌توانید با استفاده از قابلیت l2_repl– نوع سیاست جایگزینی را هنگام شبیه‌سازی برای حافظه نهان لایه دوم تعیین کنید.

**راهنمایی(بدست آوردن ترتیب دسترسی به حافظه)**

برای بدست آوردن دنباله دسترسی به حافظه، باید فایل configs/common/CacheConfig.py را تغییر دهیم. در تابع config_cache و در زیر شرط if options.l2cache ابتدا دو خط زیر را کامنت کنید:

```
system.l2.cpu_side = system.tol2bus.mem_side_ports
system.l2.mem_side = system.membus.cpu_side_ports
```

سپس درست زیر این دو خط کامنت شده، قطعه کد زیر را اضافه کنید:

```
system.monitor2 = CommMonitor()
system.monitor2.trace = MemTraceProbe(trace_file = "CT_mon2.trc.gz")
system.monitor2.slave = system.l2.mem_side

system.membus.slave = system.monitor2.master
system.l2.cpu_side = system.tol2bus.master
```

با اضافه کردن این قطعه کد شما پکت‌هایی را که روی حافظه‌نهان لایه دوم جابه‌جا می‌شوند، مانیتور می‌کنید. با اجرای شبیه‌سازی، یک فایل فشرده در آدرس m5out/CT_mon2.trc.gz ایجاد می‌شود که دارای یک فایل، حاوی اطلاعات مانیتور شده روی حافظه‌نهان لایه دوم است. شما به واسطه اسکریپت موجود در مخزن gem5 می‌توانید اطلاعات موجود در فایل مدنظر را decode کنید.

```
python3 util/decode_packet_trace.py CT_mon2.trc result.csv
```

فایل خروجی تولید شده فرمتی به صورت زیر دارد:

```
5,u,217728,128,256,0
6,u,331264,128,98,1500
5,u,221184,128,256,3000
6,u,331136,128,98,11000
6,u,323072,128,131171,15500
6,u,322176,128,99,18500
6,u,318080,128,99,22500
6,u,319104,128,99,23500
5,u,221312,128,256,24500
6,u,322304,128,99,25000
```

در هر خط یک Cache Access نمایش داده شده است که مقدار سوم آدرس خانه مدنظر از حافظه را نمایش می‌دهد. شما به واسطه این داده‌ها می‌توانید نرخ برخورد حالت بهینه را بدست آورید.

با توجه به راهنمایی های داده شده، قابلیت سیاست حافظه ی نهان را به ۵gem اضافه کردم و سپس با دو سیاست گفته شده آنها را اجرا و گزارش miss-rate را استخراج کردم، همچنین، دنباله ی دسترسی به حافظه را نیز با دنبال کردن راهنمایی ها، بدست آوردم. برنامه ی مورد بررسی:

```c
// C program to multiply two square matrices.
#include <stdio.h>
#define N 4

// This function multiplies mat1[][] and mat2[][],
// and stores the result in res[][]
void multiply(int mat1[][N], int mat2[][N], int res[][N])
{
    int i, j, k;
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            res[i][j] = 0;
            for (k = 0; k < N; k++)
                res[i][j] += mat1[i][k]*mat2[k][j];
        }
    }
}

int main()
{
    int mat1[N][N] = { {1, 1, 1, 1},
                       {2, 2, 2, 2},
                       {3, 3, 3, 3},
                       {4, 4, 4, 4}};

    int mat2[N][N] = { {1, 1, 1, 1},
                       {2, 2, 2, 2},
                       {3, 3, 3, 3},
                       {4, 4, 4, 4}};

    int res[N][N]; // To store result
    int i, j;
    multiply(mat1, mat2, res);

    return 0;
}
```

با اجرای این برنامه بر روی شبیه ساز به نتایج زیر برای miss rate با سیاست های مختلف رسیدیم.

آ) miss rate با سیاست LRU:

```
command line: ./build/X86/gem5.opt configs/deprecated/example/se.py -c binary/bench --caches --l2cache --l2_size=4kB --mem-type=DDR4_2400_16x4 --cacheline_size 128 --l2_repl=LRURP

warn: The `get_runtime_isa` function is deprecated. Please migrate away from using this function.
warn: The se.py script is deprecated. It will be removed in future releases of  gem5.
warn: The `get_runtime_isa` function is deprecated. Please migrate away from using this function.
warn: monitor2.slave is deprecated. `slave` is now called `cpu_side_port`
warn: monitor2.master is deprecated. `master` is now called `mem_side_port`
Global frequency set at 1000000000000 ticks per second
warn: failed to generate dot output from m5out/config.dot
src/mem/dram_interface.cc:690: warn: DRAM device capacity (32768 Mbytes) does not match the address range assigned (512 Mbytes)
src/base/statistics.hh:279: warn: One of the stats is a legacy stat. Legacy stat is a stat that does not belong to any statistics::Group. Legacy stat is deprecated.
system.remote_gdb: Listening for connections on port 7000
**** REAL SIMULATION ****
src/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
src/sim/mem_state.cc:443: info: Increasing stack size by one page.
src/sim/syscall_emul.cc:74: warn: ignoring syscall set_robust_list(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall rseq(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
Exiting @ tick 106625000 because exiting with last active thread context
root@DESKTOP-9EGNJVN:~/gem5/gem5# grep 'l2.overallMissRate::cpu.data' m5out/stats.txt
system.l2.overallMissRate::cpu.data         0.993168                       # miss rate for overall accesses (Ratio)
```

(ب) miss rate با سیاست FIFO:

```
command line: ./build/X86/gem5.opt configs/deprecated/example/se.py -c binary/bench --caches --l2cache --l2_size=4kB --mem-type=DDR4_2400_16x4 --cacheline_size 128 --l2_repl=FIFORP

warn: The `get_runtime_isa` function is deprecated. Please migrate away from using this function.
warn: The se.py script is deprecated. It will be removed in future releases of  gem5.
warn: The `get_runtime_isa` function is deprecated. Please migrate away from using this function.
warn: monitor2.slave is deprecated. `slave` is now called `cpu_side_port`
warn: monitor2.master is deprecated. `master` is now called `mem_side_port`
Global frequency set at 1000000000000 ticks per second
warn: failed to generate dot output from m5out/config.dot
src/mem/dram_interface.cc:690: warn: DRAM device capacity (32768 Mbytes) does not match the address range assigned (512 Mbytes)
src/base/statistics.hh:279: warn: One of the stats is a legacy stat. Legacy stat is a stat that does not belong to any statistics::Group. Legacy stat is deprecated.
system.remote_gdb: Listening for connections on port 7000
**** REAL SIMULATION ****
src/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
src/sim/mem_state.cc:443: info: Increasing stack size by one page.
src/sim/syscall_emul.cc:74: warn: ignoring syscall set_robust_list(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall rseq(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
Exiting @ tick 106625000 because exiting with last active thread context
root@DESKTOP-9EGNJVN:~/gem5/gem5# grep 'l2.overallMissRate::cpu.data' m5out/stats.txt
system.l2.overallMissRate::cpu.data          0.993168                       # miss rate for overall accesses (Ratio)
```

(ج) miss rate با سیاست Random:

```
command line: ./build/X86/gem5.opt configs/deprecated/example/se.py -c binary/bench --caches --l2cache --l2_size=4kB --mem-type=DDR4_2400_16x4 --cacheline_size 128 --l2_repl=RandomRP

warn: The `get_runtime_isa` function is deprecated. Please migrate away from using this function.
warn: The se.py script is deprecated. It will be removed in future releases of  gem5.
warn: The `get_runtime_isa` function is deprecated. Please migrate away from using this function.
warn: monitor2.slave is deprecated. `slave` is now called `cpu_side_port`
warn: monitor2.master is deprecated. `master` is now called `mem_side_port`
Global frequency set at 1000000000000 ticks per second
warn: failed to generate dot output from m5out/config.dot
src/mem/dram_interface.cc:690: warn: DRAM device capacity (32768 Mbytes) does not match the address range assigned (512 Mbytes)
src/base/statistics.hh:279: warn: One of the stats is a legacy stat. Legacy stat is a stat that does not belong to any statistics::Group. Legacy stat is deprecated.
system.remote_gdb: Listening for connections on port 7000
**** REAL SIMULATION ****
src/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
src/sim/mem_state.cc:443: info: Increasing stack size by one page.
src/sim/syscall_emul.cc:74: warn: ignoring syscall set_robust_list(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall rseq(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
src/sim/syscall_emul.cc:74: warn: ignoring syscall mprotect(...)
Exiting @ tick 106625000 because exiting with last active thread context
root@DESKTOP-9EGNJVN:~/gem5/gem5# grep 'l2.overallMissRate::cpu.data' m5out/stats.txt
system.l2.overallMissRate::cpu.data          0.999146                       # miss rate for overall accesses (Ratio)
```

سپس کدی نوشته تا با استفاده از دنباله ی دسترسی به حافظه، مقدار miss rate بهینه را یافتیم.
کد مورد نظر به این صورت است:

```python
import numpy as np
from collections import deque
import itertools


def parse_addresses(file_path):
    addresses = []
    with open(file_path, 'r') as file:
        lines = file.readlines()
        for line in lines:
            elements = line.strip().split(',')
            address = int(elements[2], 10ب)
            addresses.append(address)
    return addresses


def simulate_cache(file_path):
    addresses = parse_addresses(file_path)
    cache_simulator = CacheSimulator()

    for address in addresses:
        cache_simulator.access_count += 1
        if cache_simulator.access_cache(address):
            set_index = cache_simulator.get_set_index(address)
            cache_set = cache_simulator.cache[set_index]
            cache_simulator.future_addresses = addresses[cache_simulator.access_count:]
            best_cache_state, best_miss_count = cache_simulator.branch_and_replace(address, cache_set)
            cache_simulator.cache[set_index] = deque(best_cache_state, maxlen=cache_simulator.associativity)

    miss_rate = cache_simulator.miss_count / cache_simulator.access_count if cache_simulator.access_count > 0 else 0
    return miss_rate


miss_rate = simulate_cache('result.csv')
print(f"Miss Rate: {miss_rate:.4f}")
```

عملکرد آن به این صورت است که ابتدا آدرس ها را از فایل خوانده و سپس تک تک آدرس ها را شبیه سازی می کند، و هرگاه نیاز به جایگذاری بود، در هر دو حالت به آدرس های پیش رو نگاه می کند و تصمیم می گیرد که کدام حالت miss کمتری می خورد. البته در این جلونگری نیز ممکن است نیاز به جایگذاری بیشتری باشد که در آن صورت پیچیدگی محاسباتی الگوریتم برابر با $۲^{\text{number of replacements}}$ است که اصلا قابل محاسبه نیست پس با در نظر گرفتن این تقریب می توانیم محاسبات را سریع تر کینم.

```python
class CacheSimulator:
    def __init__(self, cache_size_kb=4, line_size_bytes=64, associativity=2):
        self.future_addresses = None
        self.cache_size_kb = cache_size_kb
        self.line_size_bytes = line_size_bytes
        self.associativity = associativity
        self.num_lines = (cache_size_kb * 1024)
        self.num_sets = self.num_lines
        self.cache = [deque(maxlen=associativity) for _ in range(self.num_sets)]
        self.access_count = 0
        self.miss_count = 0

    def get_set_index(self, address):
        return self.get_block_index(address) % self.num_sets

    def get_block_index(self, address):
        return address

    def access_cache(self, address):
        set_index = self.get_set_index(address)
        block_index = self.get_block_index(address)
        cache_set = self.cache[set_index]

        if block_index in cache_set:
            return False
        else:
            self.miss_count += 1
            if len(cache_set) >= self.associativity:
                return True
            cache_set.append(block_index)
            return False

    def branch_and_replace(self, address, cache_set):
        block_index = self.get_block_index(address)
        best_miss_count = float('inf')
        best_cache_state = None

        for i in range(len(cache_set)):
            new_cache_set = cache_set.copy()
            new_cache_set[i] = block_index
            miss_count = self.simulate_future_accesses(new_cache_set)
            if miss_count < best_miss_count:
                best_miss_count = miss_count
                best_cache_state = new_cache_set

        return best_cache_state, best_miss_count

    def simulate_future_accesses(self, cache_set):
        future_miss_count = 0
        for address in self.future_addresses:
            block_index = self.get_block_index(address)
            if block_index not in cache_set:
                future_miss_count += 1
        return future_miss_count
```

داخل این کلاس نیز توابعی تعبیه شده تا مشخص کنند، دسترسی به یک حافظه باعث جایگذاری می شود یا خیر و همچنین محاسبه ی miss های آینده

```
Miss Rate: 0.9641
```

در انتها نیز miss rate بهینه برابر این مقدار است.