# Homework 4

Huy Quang Lai
132000359

*Texas A&M University*

8 December 2022

An Aggie does not lie, cheat or steal.
Nor does an Aggie tolerate those who do.

---

## Chapter 4

### 4.13

Correct

```
Fetch:
icode:ifun <- M1[0x016] = 0x3:0x0
rA:rB <- M1[0x017] = 0xf, 0x4
valC <- M8[0x018] = 0x0000000000000080
valP <- 0x016 + 0x0A = 0x20

Decode:
do nothing

Execute:
valE = 0x0 + 0x0000000000000080 = 0x0000000000000080

Memory:
do nothing

Write Back:
R[%rsp] <- valE = 0x0000000000000080

PC Update:
PC <- valP = 0x020
```

## 4.14

Correct

```
Fetch:
icode:ifun <- M1[0x02c] = b:0
rA:rB <- M1[0c02b] = 0:f
valP <- 0x02c + 0x2 = 0x02e

Decode:
valA <- R[%rsp]= 0x78
valB <- R[%rsp]= 0xC

Execute
valE <- 120 + 8 = 0x80
valM <- M8[120]= 0x9

Write Back
R[%rsp] <- 0x80
R[%rax] <- 0x9

PC Update:
PC <- 0x02e
```

## 4.43

Correct
Instruction Frequency: 0.20
Cost of misprediction: 2
Rate of misprediction: 0.35 $CPI = 0.20 * 0.35 * 2 = 1.25$

## 6.12

Correct

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CT | CT | CT | CT | CT | CT | CT | CT | CI | CI | CI | CO | CO |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

## 6.17

Correct

```
             tag    si off
s[0][0] = 0b0000 0 000
s[0][1] = 0b0000 0 100
s[1][0] = 0b0000 1 000
s[1][1] = 0b0000 1 100
d[0][0] = 0b0001 0 000
d[0][1] = 0b0001 0 100
d[1][0] = 0b0001 1 000
d[1][1] = 0b0001 1 100
```

Given
Cache: | `d[0][0]` | nothing |

| src arr | | | dst arr | | |
|---|---|---|---|---|---|
| | Col 0 | Col 1 | | Col 0 | Col 1 |
| Row 0 | m | | Row 0 | m | |
| Row 1 | | | Row 1 | | |

---

Read `s[0][1]`
Cache: | `s[0][1]` | nothing |

| src arr | | | dst arr | | |
|---|---|---|---|---|---|
| | Col 0 | Col 1 | | Col 0 | Col 1 |
| Row 0 | m | m | Row 0 | m | |
| Row 1 | | | Row 1 | | |

---

Write `d[1][0]`
Cache: | `s[0][1]` | `d[1][0]` |

| src arr | | | dst arr | | |
|---|---|---|---|---|---|
| | Col 0 | Col 1 | | Col 0 | Col 1 |
| Row 0 | m | m | Row 0 | m | |
| Row 1 | | | Row 1 | m= | |

---

Read `s[1][0]`
Cache: | `s[0][1]` | `s[1][0]` |

| src arr | | | dst arr | | |
|---|---|---|---|---|---|
| | Col 0 | Col 1 | | Col 0 | Col 1 |
| Row 0 | m | m | Row 0 | m | |
| Row 1 | m | | Row 1 | m | |

---

Write `d[0][1]`

Cache: | `d[0][1]` | `s[1][0]` |

src arr dst arr

| | Col 0 | Col 1 | | Col 0 | Col 1 |
|---|---|---|---|---|---|
| Row 0 | m | m | Row 0 | m | m |
| Row 1 | m | | Row 1 | m | |

src arr dst arr

| | Col 0 | Col 1 | | Col 0 | Col 1 |
|---|---|---|---|---|---|
| Row 0 | m | m | Row 0 | m | |
| Row 1 | m | | Row 1 | m | |

---

Read `s[1][1]`

Cache: | `d[0][1]` | `s[1][0]` |

src dst arr

| | Col 0 | Col 1 | | Col 0 | Col 1 |
|---|---|---|---|---|---|
| Row 0 | m | m | Row 0 | m | m |
| Row 1 | m | h | Row 1 | m | |

src arr dst arr

| | Col 0 | Col 1 | | Col 0 | Col 1 |
|---|---|---|---|---|---|
| Row 0 | m | m | Row 0 | m | |
| Row 1 | m | | Row 1 | m | |

---

Write `d[1][1]`

Cache: | `d[1][0]` | `d[1][1]` |

src arr dst arr

| | Col 0 | Col 1 | | Col 0 | Col 1 |
|---|---|---|---|---|---|
| Row 0 | m | m | Row 0 | m | m |
| Row 1 | m | h | Row 1 | m | m |

```
            tag   si  off
s[0][0]  = 0b000 00  000
s[0][1]  = 0b000 00  100
s[1][0]  = 0b000 01  000
s[1][1]  = 0b000 01  100
d[0][0]  = 0b000 10  000
d[0][1]  = 0b000 10  100
d[1][0]  = 0b000 11  000
d[1][1]  = 0b000 11  100
```

```
Read  s[0][0] Miss cache[0][0] = s[0][0], s[0][1]
Write d[0][0] Miss cache[1][0] = d[0][0], d[0][1]
Read  s[0][1] Hit  read cache[0][1] offset ob100
Write d[1][0] Miss cache[1][1] = d[1][0], d[1][1]
Read  s[1][0] Miss cahce[0][1] = s[1][0], s[1][1]
Write d[0][1] Hit  read cache[1][0] offset 0b100
Read  s[1][1] Hit  read cache[1][1] offset 0b100
Write d[1][1] Hit  read cache[1][1] offset 0b100
```

| Cache: | | | | |
|---|---|---|---|---|
| | `s[0][0]` | `s[1][0]` | `d[0][0]` | `d[1][0]` |

## 6.18

Correct
Each 32-byte cache block holds two consecutive `algae_position` structures.
With cache being 2048-bytes, it can hold 64 blocks.
Each `algae_position` in `grid` is read twice, once for its x-coordinate and once for its y-coordinate.
Since the for loop traverses the entire grid, there are a total of $32 \times 32 \times 2 = 2048$ reads.
Since two consecutive `algae_position` structures are written every block, the cache pattern will miss, then hit and alternate between these states through the entire `grid`.
Therefore, the total number of misses would be $1024$ and the miss rate is $50\%$.