CENG 3420 Midterm (2021 Spring)

Name: Ng Chi Hon
ID: 1155116317

Q0 (0 marks)

1. What is your last digit of your SID (0 is regarded as 10)? This value is defined as `NUM_1` in the whole question paper.
2. What is your last two digits of your SID (00 is regarded as 100)? This value is defined as `NUM_2` in the whole question paper.
3. What is your last three digits of your SID? This value is defined as `NUM_3` in the whole question paper.

Example: if your SID is 12345678, then $\texttt{NUM\_1} = 8, \texttt{NUM\_2} = 78, \texttt{NUM\_3} = 678$.

Q1 (10 marks)

Select and fill the correct answer.

1. RISC-V is a ____ and ____ ISA.

   A. little-endian RISC

   B. little-endian CISC

   C. big-endian RISC

   D. big-endian CISC

2. Virtual memory systems use ____ mechanism.

   A. write-back

   B. write-through

   C. write-allocate

3. The largest positive integer in IEEE754 single-precision floating-point format is ____

   A. $2^{126} - 2^{103}$

   B. $2^{127} - 2^{104}$

   C. $2^{127} - 2^{103}$

   D. $2^{128} - 2^{104}$

4. ____ is a type of single address space multiprocessor in which some memory accesses are much faster than others depending on which processor asks for which word.

   A. UMA

   B. NUMA

5. Dividing 10101_2 by 11_2, the quotient is ____ and the reminder is ____.

   A. 100_2 01_2

   B. 111_2 01_2

   C. 111_2 0_2

   D. 100_2 0_2

6. ____ stores data as electric charge on a capacitor.

   A. SRAM

   B. DRAM

Q2 (5 marks)

1. Describe the steps that transform a program written in a high-level language such as C into a representation that is directly executed by a computer processor.
2. Write C statements that corresponds to the two RISC-V assembly instructions below.

```
srli t0, t1, 3
andi t0, t2, 0xFEF
```

   You should define variables that hold corresponding register values explicitly.

Q3 (10 marks)

Consider a RISC-V code snippet, and assume that the code is running on a 5-stage RISC-V core.

```
add x15, x12, x11
ld x13, 4(x15)
ld x12, 0(x2)
or x13, x15, x13
sd x13, 0(x15)
```

1. Find all data dependences in this instructioin sequence.
2. If the design has not implemented any forwarding or hazard detection, insert NOPs to ensure the correct execution.
3. If there is forwarding, for the first seven cycles during the execution of this code, specif which signals are asserted in each cycle by hazard detection and forwarding units.

Q4 (15 marks)

We have a loop as follows.

```
LOOP: ld a0, 0(a3)
      ld a1, 8(a3)
      add a2, a0, a1
      subi a3, a3, 16
      bnez a2, LOOP
```

Assume that we have the perfect branch prediction (i.e., no stalls due to control hazards) and there are no delay slots that the pipeline has full forwarding support, and that branches are resolved in the EX (as opposed to the ID) stage.

1. Show a pipeline execution diagram for the first two iterations of this loop.
2. Mark pipeline stages that do not perform useful work. How often while the pipeline is full do we have a cycle in which all five pipeline stages are doing useful work? (Begin with the cycle during which the `subi` is in the IF stage. End with the cycle during which the `bnez` is in the IF stage)

Q5 (15 marks)

Consider the following loops written in C programming language:

```
for(int i = 0; i < a; i++)
    for(int j = 0; j < b; j++)
        A[j * 4] = i + j;
```

The values of `a`, `b`, `i` and `j` are in registers `t0`, `t1`, `t2` and `t3`, respectively. The register `a0` holds the base address of the array `A`. Translate to the corresponding RISC-V assembly code.

Q6 (15 marks) There is a computer that has a 64MB byte-addressable main memory. Instructions and data are stored in separated caches, each of which has eight 64B cache lines. The data cache uses **direct-mapping**. Now there are two programs in the following form.

Program A:

```
int a[64][64];
int sum_array1() {
    int i, j, sum = 0;
    for(i = 0; i < 64; i++)
        for(j = 0; j < 64; j++)
            sum += a[i][j];
    return sum;
}
```

Program B:

```
int a[64][64];
int sum_array1() {
    int i, j, sum = 0;
    for(j = 0; j < 64; j++)
        for(i = 0; i < 64; i++)
            sum += a[i][j];
    return sum;
}
```

Suppose `int` data is represented in 32-bit 2's complement and `i`, `j`, `sum` are stored in specific registers. Arrays are stored in row-major with the start address NUM_3_{10} (i.e., NUM_3 in decimal) in the main memory. Answer the following questions.

1. What are the line numbers of the main memory blocks that contain `a[0][31]` and `a[2][2]` respectively? (Cache line number starts from 0)
2. What are the data cache hit rates of program A and B?

Q7 (5 marks)

Describe two cache replacement strategies.

1. Write-through
2. Write-back

Q8 (15 marks)

The following table shows the different instructions ratios

Table 1: Instruction mix

R-type	I-type (non-Id)	Load	Store	Branch	Jump
$2 \times$ NUM_1	25%	22%	8%	$45\% - 3 \times$ NUM_1	NUM_1

1. What fraction of all instructions use the data memory?
2. What fraction of all instructions use the instruction memory?
3. What fraction of all instructions use the sign extend?
4. What is the sign extend doing during cycles in whih its output is not needed?

Q9 (10 marks)

Given the following specifications of the datapath latencies:

Table 2: Specification of the datapath latencies (unit: *ps*)

Stages	IF	ID	EX	MEM	WB
Latencies (ps)	NUM_2	NUM_3	550	800	600

1. What is the clock cycle time in a pipelined and non-pipelined processor?
2. What is the total latency of an `lw` instruction in a pipelined and non-pipelined processor?
3. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

Q2 (5 marks)

1. Describe the steps that transform a program written in a high-level language such as C into a representation that is directly executed by a computer processor.
2. Write C statements that corresponds to the two RISC-V assembly instructions below.

```
srli t0, t1, 3
```
$t_0 = t_1 \times 2^3$

```
andi t0, t2, 0xFEF
```
$t_0 =$

You should define variables that hold corresponding register values explicitly.

1) C program will compile to Assembly language program with Compiler, Then Assembly language will turn to machine code that can read from processor by assembler.

2) $t0 = t_1 \times 2 \times 2 \times 2$

$t_0 = t2$ && FEF

Q3 (10 marks)

Consider a RISC-V code snippet, and assume that the code is running on a 5-stage RISC-V core.

```
add x15, x12, x11
ld x13, 4(x15)
ld x12, 0(x2)
or x13, x15, x13
sd x13, 0(x15)
```

1. Find all data dependences in this instructioin sequence.
2. If the design has not implemented any forwarding or hazard detection, insert NOPs to ensure the correct execution.
3. If there is forwarding, for the first seven cycles during the execution of this code, specif which signals are asserted in each cycle by hazard detection and forwarding units.

1)

```
add x15, x12, x11
ld  x13, 4(x15)
ld  x12, 0(x2)
or  x13, x15, x13
sd  x13, 0(x15)
```

← x15 as 3 stage after load write, directly read from register
← x15 as 4 stage after

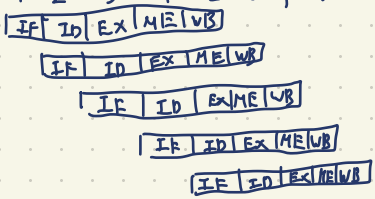
2)

```
add x15, x12, x11
NOP
NOP
ld  x13, 4(x15)
ld  x12, 0(x2)
NOP
or  x13, x15, x13
NOP
NOP
sd  x13, 0(x15)
```

3)

```
add x15, x12, x11
ld  x13, 4(x15)
ld  x12, 0(x2)
or  x13, x15, x13
sd  x13, 0(x15)
```

ForwardA	X	X	00	10	01	00	00	XX	XX
ForwardB	X	X	00	00	01	10	10	XX	XX
Cycle	1	2	3	4	5	6	7	8	9



Q4 (15 marks)

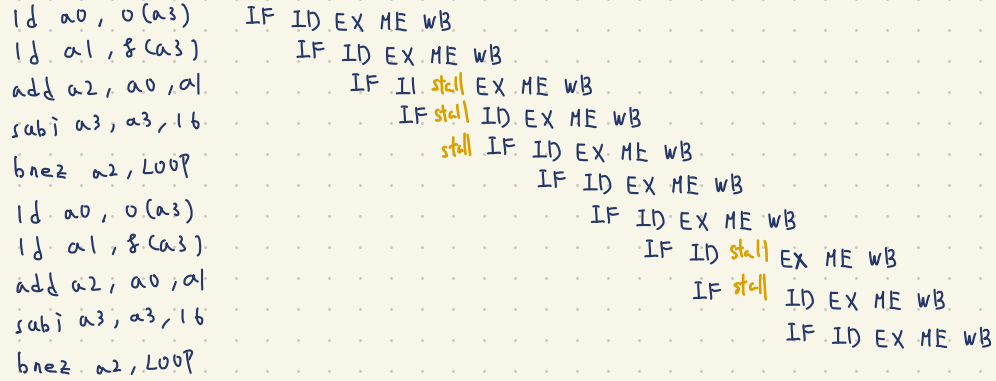
We have a loop as follows.

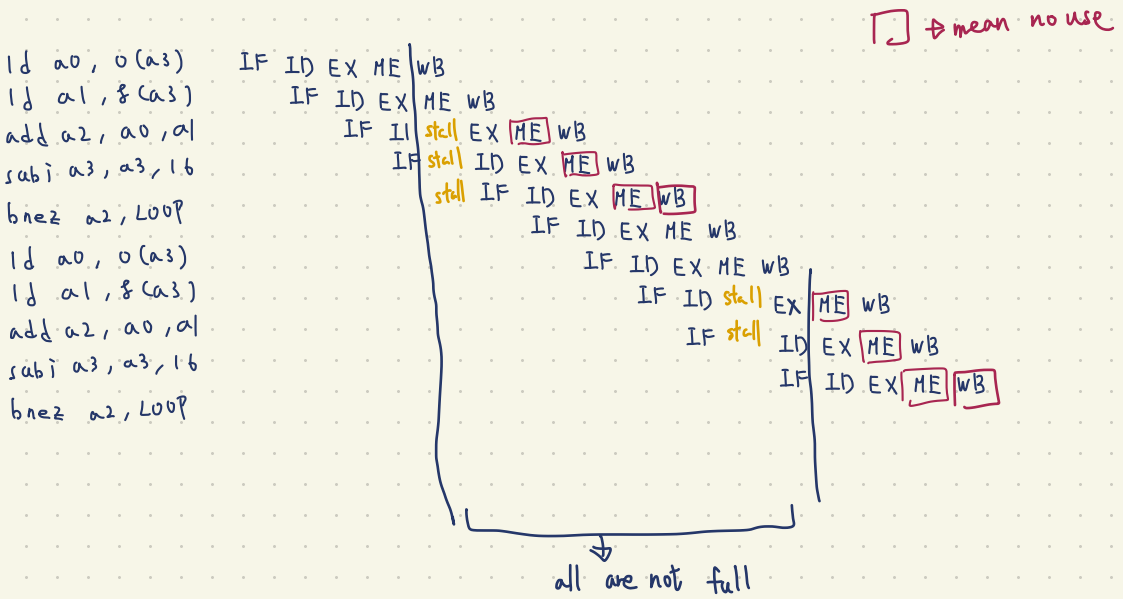
```
LOOP: ld a0, 0(a3)
      ld a1, 8(a3)
      add a2, a0, a1
      subi a3, a3, 16
      bnez a2, LOOP
```

Assume that we have the perfect branch prediction (i.e., no stalls due to control hazards) and there are no delay slots that the pipeline has full forwarding support, and that branches are resolved in the EX (as opposed to the ID) stage.

1. Show a pipeline execution diagram for the first two iterations of this loop.
2. Mark pipeline stages that do not perform useful work. How often while the pipeline is full do we have a cycle in which all five pipeline stages are doing useful work? (Begin with the cycle during which the `subi` is in the IF stage. End with the cycle during which the `bnez` is in the IF stage)

Q1





There is no full pipeline that all stage do useful work. some stall, or some is not doing useful work,

Q5 (15 marks)

Consider the following loops written in C programming language:

```
for(int i = 0; i < a; i++)
    for(int j = 0; j < b; j++)
        A[j * 4] = i + j;
```

The values of `a`, `b`, `i` and `j` are in registers `t0`, `t1`, `t2` and `t3`, respectively. The register `a0` holds the base address of the array `A`. Translate to the corresponding RISC-V assembly code.

t0 = a t1 = b t2 = i t3 = j

a0

```
LOOP1:  bge  t2, t0, Exit
        addi t2, t2, 1
        j    LOOP1

LOOP2:  bge  t3, t1, LOOP1
        sll  t4, t3, 2      # t4 as temp, t4 = j x 4
        sll  t4, t4, 2      # address x4 for 1 word
        add  t5, t2, t3
        sw   t5, 0(t4)

        addi t3, t3, 1
        j    LOOP2

Exit:
```

Q6 (15 marks) There is a computer that has a 64MB byte-addressable main memory. Instructions and data are stored in separated caches, each of which has eight 64B cache lines. The data cache uses **direct-mapping**. Now there are two programs in the following form.

Program A:

```
int a[64][64];
int sum_array1() {
    int i, j, sum = 0;
    for(i = 0; i < 64; i++)
        for(j = 0; j < 64; j++)
            sum += a[i][j];
    return sum;
}
```

Program B:

```
int a[64][64];
int sum_array1() {
    int i, j, sum = 0;
    for(j = 0; j < 64; j++)
        for(i = 0; i < 64; i++)
            sum += a[i][j];
    return sum;
}
```

Suppose `int` data is represented in 32-bit 2's complement and `i`, `j`, `sum` are stored in specific registers. Arrays are stored in row-major with the start address NUM_3$_{10}$ (i.e., NUM_3 in decimal) in the main memory. Answer the following questions.

1. What are the line numbers of the main memory blocks that contain `a[0][31]` and `a[2][2]` respectively? (Cache line number starts from 0)
2. What are the data cache hit rates of program A and B?

Q7 (5 marks)

Describe two cache replacement strategies.

1. Write-through
2. Write-back

1) Write-through, write the data into both cache block & next level in memory hierarchy

2) Write-back, write the data only into cache block when cache block is "evicted", then write data back to memory hierarchy

Q8 (15 marks)

The following table shows the different instructions ratios

Table 1: Instruction mix

R-type	I-type (non-Id)	Load	Store	Branch	Jump
$2 \times$ NUM_1	25%	22%	8%	$45\% - 3 \times$ NUM_1	NUM_1

1. What fraction of all instructions use the data memory?
2. What fraction of all instructions use the instruction memory?
3. What fraction of all instructions use the sign extend?
4. What is the sign extend doing during cycles in whih its output is not needed?

1) 22% + 8% = 30%

2) 100% (every instruction must fetch from instruction memory before it can executed)

3) 25% + 22% + 8% + 45% - 3×NUM_1 + NUM_1
= 100% - 2NUM_1 = 100% - 2·7 = 84%

4) The sign extend alway output during every cycle. If output is not needed, the system ignore it.

Q9 (10 marks)

Given the following specifications of the datapath latencies:

Table 2: Specification of the datapath latencies (unit: *ps*)

Stages	IF	ID	EX	MEM	WB
Latencies (ps)	NUM_2	NUM_3	550	800	600

17 317

1. What is the clock cycle time in a pipelined and non-pipelined processor?
2. What is the total latency of an `lw` instruction in a pipelined and non-pipelined processor?
3. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

1) Non-pipelined : 2284 ps pipelined : 800 ps

2) Non-pipelined: 2284ps pipelined : 4000ps

3) Split MEM stage, then new clock cycle time is 600ps