# Homework 1

**Hamza Kamal**

*February 10, 2025*

### Question:

(2pts) Register Transfer Language (RTL)
Trace the data flow of the following code using RTL. Include the first instruction fetch. LDR R2, [R1]

### Answer:

$MAR \leftarrow PC$
$PC \leftarrow PC + 2$
$IR \leftarrow mem[MAR]$
$MAR \leftarrow R1$
$R2 \leftarrow mem[MAR]$

---

### Question:

(2 pts) Application Program Status Register (APSR)'s flags
After the following piece of instructions is executed, what value will be maintained in each of NZCV flags in APSR?

### Answer:

| Flag | Value |
|---|---|
| N | 1 |
| Z | 0 |
| C | 0 |
| V | 1 |

---

## Question:

(4 pts) Memory Endianness and Alignment
1. As you see the following example with #1234 at memory address 0x20000000, allocate #1234567890 to memory address 0x20001000. (2pts)
An example:
Big Endian:

| Address | Data Contents (in hex) |
|---|---|
| 0x20000000 | 04 |
| 0x20000001 | D2 |
| 0x20000002 | |
| 0x20000003 | |

Little Endian:

| Address | Data Contents (in hex) |
|---|---|
| 0x20000000 | D2 |
| 0x20000001 | 04 |
| 0x20000002 | |
| 0x20000003 | |

## Answer:

#1234567890
#1234567890 in hex = $0x499602D2_{16}$
Big Endian:

| Address | Data Contents (in hex) |
|---|---|
| 0x20000000 | 49 |
| 0x20000001 | 96 |
| 0x20000002 | 02 |
| 0x20000003 | D2 |

Little Endian:

| Address | Data Contents (in hex) |
|---|---|
| 0x20000000 | D2 |
| 0x20000001 | 02 |
| 0x20000002 | 96 |
| 0x20000003 | 49 |

---

## Question:

As you see the following example with exampleData, allocate myData to the memory and fill out the spaces to indicate how each data element is mapped. Assume that the memory is based on a 32-bit addressing system. (2pts)

An Example:

```
struct exampleData {
   char a;
   short b;
}
```

| | + 0th | + 1st | + 2nd | + 3rd |
|---|---|---|---|---|
| 0th byte | a | | b | b |
| 4th byte | | | | |
| 8th byte | | | | |

## Answer:

```
struct myData {
   char a;
   long long int b;
   double c;
   short d;
```

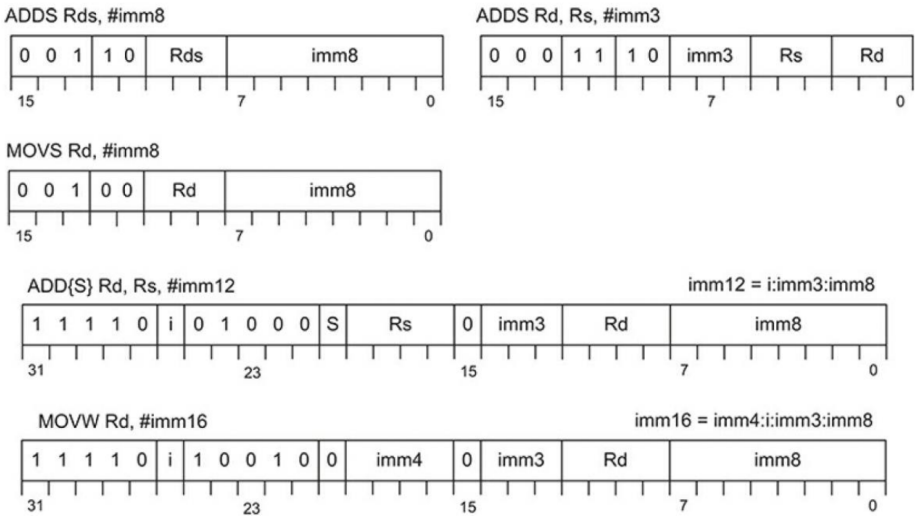
```
   char *e;
   float f;
}
```

| | + 0th | + 1st | + 2nd | + 3rd |
|---|---|---|---|---|
| 0th byte | a | | b | b |
| 4th byte | b | b | b | b |
| 8th byte | b | b | | c |
| 12th byte | c | c | c | c |
| 16th byte | c | c | c | |
| 20th byte | d | d | | e |
| 24th byte | | f | f | f |
| 28th byte | f | | | |
| 32nd byte | | | | |

---

## Question:

(4pts) Assemble the codes Using the following picture of Thumb2's encoding formats, convert the following three assembly language instructions to the machine codes in hexadecimal. You have to show your work, otherwise you get zero.




1. ADDS R7, R5, #7 (2pts)

## Answer:

$R5 = 101$
$R7 = 111$
$7 = 111_2$

| 000 | 11 | 10 | imm3 | Rs | Rd |
|---|---|---|---|---|---|
| 000 | 11 | 10 | 111 | 101 | 111 |

---

## Question:

2. MOVW R10, #0x1234. (2pts)

## Answer:

$R10 = 1010$
$0x1234 = 0001001000110100$

| 1111 | i | 10010 | 0 | imm4 | 0 | imm3 | Rd | imm8 |
|---|---|---|---|---|---|---|---|---|
| 1111 | 0 | 10010 | 0 | 0010 | 0 | 010 | 1010 | 0011 0100 |

---