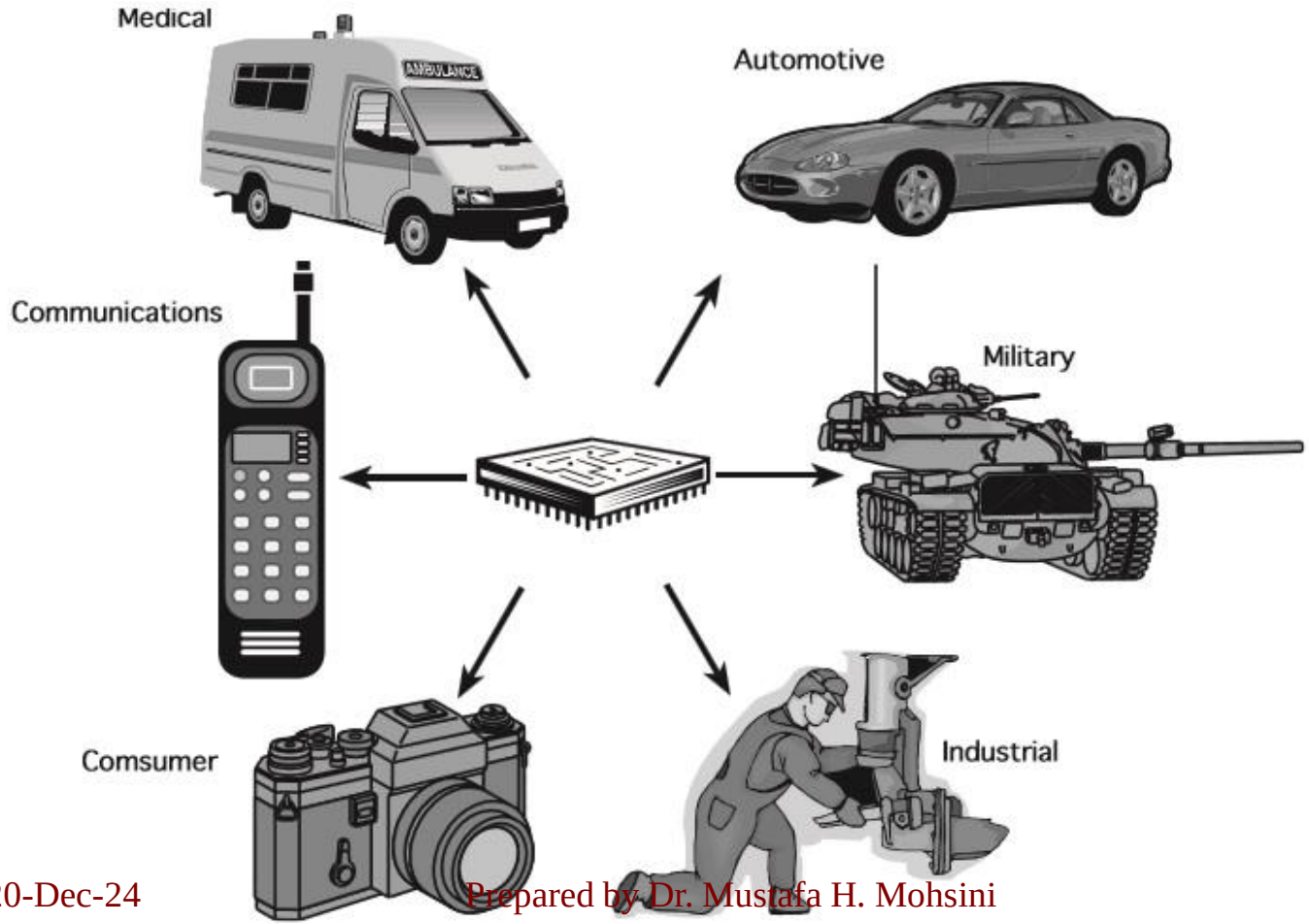
CT 411 Embedded Systems Examples on Lecture #3



:Example 1: Simple “if” Statement

C programming

```c
if (x > 0) {
    y = x;
}
```

Corresponding ARM code

```
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    CMP R1, #0          // Compare x (R1) with 0
    BLE end_if          // If x <= 0, skip to the end
    LDR R2, =y          // Load address of y into R2
    STR R1, [R2]        // Store x into y

end_if:
    B exit              // End the program

exit:
```

Example 2: “if-else” Statement

C programming

```c
if (x > 0) {
    y = x;
} else {
    y = -x;
}
```

Corresponding ARM code

```
    LDR R0, =x              // Load address of x into R0
    LDR R1, [R0]            // Load value of x into R1
    CMP R1, #0              // Compare x (R1) with 0
    BGT positive_case       // If x > 0, branch to positive_case
    NEG R2, R1              // Else, calculate -x
    B store_y               // Skip to store_y

positive_case:
    MOV R2, R1              // y = x

store_y:
    LDR R3, =y              // Load address of y into R3
    STR R2, [R3]            // Store result in y

exit:
```

Example 3: Nested-if Statement

C programming

```c
if (x > 0) {
    if (y > 0) {
        z = x + y;
    }
}
```

Corresponding ARM code

```
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    CMP R1, #0          // Compare x (R1) with 0
    BLE exit            // If x <= 0, skip to exit

    LDR R2, =y          // Load address of y into R2
    LDR R3, [R2]        // Load value of y into R3
    CMP R3, #0          // Compare y (R3) with 0
    BLE exit            // If y <= 0, skip to exit

    ADD R4, R1, R3      // z = x + y
    LDR R5, =z          // Load address of z into R5
    STR R4, [R5]        // Store result in z

exit:
```

Example 4: If-else if-else Statement

C programming

```c
if (x > 0) {
    if (y > 0) {
        z = x + y;
    }
}
```

Corresponding ARM code

```
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    CMP R1, #0          // Compare x (R1) with 0
    BGT set_y1          // If x > 0, branch to set_y1
    BEQ set_y0          // If x == 0, branch to set_y0
    MOV R2, #-1         // Else: y = -1
    B store_y           // Skip to store_y

set_y1:
    MOV R2, #1          // y = 1
    B store_y           // Skip to store_y

set_y0:
    MOV R2, #0          // y = 0

store_y:
    LDR R3, =y          // Load address of y into R3
    STR R2, [R3]        // Store result in y

exit:
```

Example 5: if-else Statement

C programming

```c
if (x > 0 && y > 0) {
    z = x + y;
} else {
    z = 0;
}
```

Corresponding ARM code

```
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    CMP R1, #0          // Compare x > 0
    BLE set_z0          // If x <= 0, branch to set_z0

    LDR R2, =y          // Load address of y into R2
    LDR R3, [R2]        // Load value of y into R3
    CMP R3, #0          // Compare y > 0
    BLE set_z0          // If y <= 0, branch to set_z0

    ADD R4, R1, R3      // z = x + y
    LDR R5, =z          // Load address of z into R5
    STR R4, [R5]        // Store result in z
    B exit              // Skip to exit

set_z0:
    MOV R4, #0          // z = 0
    LDR R5, =z          // Load address of z into R5
    STR R4, [R5]        // Store result in z

exit:
```

Points to Note

1. **Registers**:
   - General-purpose registers: `R0` to `R12`.
   - Temporary calculations and conditions use these registers.
2. **Branching**:
   - `B` is an unconditional branch.
   - Conditional branches: `BGT` (greater than), `BLE` (less or equal), `BEQ` (equal).
3. **Memory Access**:
   - Use `LDR` and `STR` to load and store values from/to memory.
   - Memory addresses are specified with labels like `=x`.

Example 7

C programming

```c
int x = 5, y = 3;
int z = x + y;
```

Corresponding ARM code

```
_start:
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    LDR R0, =y          // Load address of y into R0
    LDR R2, [R0]        // Load value of y into R2
    ADD R3, R1, R2      // R3 = x + y
    LDR R0, =z          // Load address of z into R0
    STR R3, [R0]        // Store result in z
    B exit
exit:
```

Example 8

C programming

```c
if (x > y) {
    z = 1;
}
```

Corresponding ARM code

```
_start:
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    LDR R0, =y          // Load address of y into R0
    LDR R2, [R0]        // Load value of y into R2
    CMP R1, R2          // Compare x (R1) with y (R2)
    BLE end_if          // If x <= y, branch to end_if
    MOV R3, #1          // Set z to 1
    LDR R0, =z          // Load address of z into R0
    STR R3, [R0]        // Store 1 in z
end_if:
    B exit
exit:
```

Points to Note

- `LDR` loads values from memory into registers.
- `ADD` adds values in registers and stores the result in `R3`.
- `STR` stores the result back into memory.
- `CMP` compares two registers.
- `BLE` branches if the comparison result is "less than or equal."
- `MOV` sets a register to a constant value.

Example 9

C programming

```c
if (x > y) {
    z = 1;
} else {
    z = 0;
}
```

Corresponding ARM code

```
_start:
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    LDR R0, =y          // Load address of y into R0
    LDR R2, [R0]        // Load value of y into R2
    CMP R1, R2          // Compare x (R1) with y (R2)
    BLE else_case       // If x <= y, branch to else_case
    MOV R3, #1          // Set z to 1
    LDR R0, =z          // Load address of z into R0
    STR R3, [R0]        // Store 1 in z
    B exit              // Skip else_case

else_case:
    MOV R3, #0          // Set z to 0
    LDR R0, =z          // Load address of z into R0
    STR R3, [R0]        // Store 0 in z

exit:
```

Example 10

C programming

```c
int i = 0;
while (i < 5) {
    i++;
}
```

Explanation:

- `CMP` compares the loop variable with 5.
- `ADD` increments the variable.
- `BGE` branches if the loop condition is no longer true.

Corresponding ARM code

```
_start:
    LDR R0, =i          // Load address of i into R0
    LDR R1, [R0]        // Load value of i into R1

loop_start:
    CMP R1, #5          // Compare i (R1) with 5
    BGE exit            // If i >= 5, exit loop
    ADD R1, R1, #1      // Increment i by 1
    STR R1, [R0]        // Store updated i back to memory
    B loop_start        // Repeat loop

exit:
```

Example 11

C programming

```c
int sum = 0;
for (int i = 0; i < 5; i++) {
    sum += i;
}
```

Corresponding ARM code

```
_start:
    LDR R0, =i          // Load address of i into R0
    LDR R1, =sum        // Load address of sum into R1
    MOV R2, #0          // Set i to 0
    MOV R3, #0          // Set sum to 0

loop_start:
    CMP R2, #5          // Compare i (R2) with 5
    BGE exit            // If i >= 5, exit loop
    ADD R3, R3, R2      // sum += i
    ADD R2, R2, #1      // Increment i
    B loop_start        // Repeat loop

exit:
    LDR R0, =sum        // Load address of sum into R0
    STR R3, [R0]        // Store result in sum
```

Example 12

- What is the value of and R1 before and after each of the following operations?

1. MOV R0, #15 // Load 15 into R0

   LSL R1, R0, #1 // Logical shift left R0 by 1, store in R1

Before: R0 = 0x00001111

After: R0 = 0x00001111

R1 = 0x00011110

Examples on LSL, LSR and ASR

- What is the value of R0 and R1 before and after each of the following operations?

1. MOV R0, #15 // Load 15 into R0

   LSL R1, R0, #1 // Logical shift left R0 by 1, store in R1

   Before: R0 = 0x00001111

   After: R0 = 0x00001111

   R1 = 0x00011110

Examples on LSL, LSR and ASR

2. MOV R1, R0, LSL #4 given that R0 = 0x00001110

   Before: R0 = 0x00001110

   After: R0 = 0x00001110

   R1 = 0x11100000

Examples on LSL, LSR and ASR

3. MOV R1, R0, LSR #3 given that R0 = 0x00010000

   Before: R0 = 0x00010000

   After: R0 = 0x00010000

   R1 = 0x00000010

4. MOV R1, R0, LSR #2 given that R0 = 0x11111111

   Before: R0 = 0x11111111

   After: R0 = 0x11111111

   R1 = 0x00111111

Examples on LSL, LSR and ASR

5. MOV R1, R0, ASR #2 given that R0 = 0x11111000

   Before: R0 = 0x11111000

   After: R0 = 0x11111000

   R1 = 0x11111110

6. MOV R1, R0, ASR #4 given that R0 = 0x0011100000000000010

   Before: R0 = 0x0011100000000000010

   After: R0 = 0x0011100000000000010

   R1 = 0x00000111000000000000

Examples on LSL, LSR and ASR and Conditional Operations

C programming

```c
if (x > y) {
    z = x << 2;  // Logical shift left by 2
}
```

Corresponding ARM code

```
_start:
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    LDR R0, =y          // Load address of y into R0
    LDR R2, [R0]        // Load value of y into R2
    CMP R1, R2          // Compare x (R1) and y (R2)
    MOVGT R3, R1, LSL #2 // If x > y, perform LSL #2 and store in R3
    LDR R0, =z          // Load address of z into R0
    STRGT R3, [R0]      // If condition is true, store R3 into z
    B exit
exit:
```

Examples on LSL, LSR and ASR and Conditional Operations - 1

C programming

```c
if (x == y) {
    z = x >> 1;  // Logical shift right by 1
}
```

Corresponding ARM code

```
_start:
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    LDR R0, =y          // Load address of y into R0
    LDR R2, [R0]        // Load value of y into R2
    CMP R1, R2          // Compare x (R1) and y (R2)
    MOVEQ R3, R1, LSR #1 // If x == y, perform LSR #1 and store i
    LDR R0, =z          // Load address of z into R0
    STREQ R3, [R0]      // If condition is true, store R3 into z
    B exit
exit:
```

Examples on LSL, LSR and ASR and Conditional Operations - 2

C programming

```c
if (x == y) {
    z = x >> 1;  // Logical shift right by 1
}
```

Corresponding ARM code

```
_start:
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    LDR R0, =y          // Load address of y into R0
    LDR R2, [R0]        // Load value of y into R2
    CMP R1, R2          // Compare x (R1) and y (R2)
    MOVEQ R3, R1, LSR #1 // If x == y, perform LSR #1 and store i
    LDR R0, =z          // Load address of z into R0
    STREQ R3, [R0]      // If condition is true, store R3 into z
    B exit
exit:
```

Examples on LSL, LSR and ASR and Conditional Operations - 3

C programming

```c
if (x < y) {
    z = x >> 2;  // Arithmetic shift right by 2
}
```

Corresponding ARM code

```
_start:
    LDR R0, =x          // Load address of x into R0
    LDR R1, [R0]        // Load value of x into R1
    LDR R0, =y          // Load address of y into R0
    LDR R2, [R0]        // Load value of y into R2
    CMP R1, R2          // Compare x (R1) and y (R2)
    MOVLT R3, R1, ASR #2 // If x < y, perform ASR #2 and store in R3
    LDR R0, =z          // Load address of z into R0
    STRLT R3, [R0]      // If condition is true, store R3 into z
    B exit
exit:
```

THE END!!