CSE 30: Computer Organization and Systems Programming

Lecture 12: ARM Assembly Data Transfer Instructions

Diba Mirza
University of California, San Diego
(Recap) Arithmetic Instructions

\[ \text{ADD} \quad r_0, r_1, r_2 \]

\[ \text{SUB} \quad r_0, r_1, #10 \]

\[ \text{MUL} \quad r_0, r_1, r_2 \]

\[ \text{Inc,} \quad r_0 = r_1 + r_2 \]

\[ r_0 = r_1 - 10 \]

\[ r_0 = r_1 + r_2 \quad (32 \text{ bits}) \]
Assignment Instructions

In C:

\[ a = b; \]  \text{Assume integers}
\[ a = 10; \]

Mapping
\[ a : r_0 \]
\[ b : r_1 \]

In ARM:

\[
\text{mov } r_0, r_1 \quad @ r_0 = r_1
\]
\[
\text{mov } r_0, \#10 \quad @ r_0 = r_1
\]
Data transfer (memory) Instructions

- Separate instructions to transfer data between registers and memory:
  - Memory to register (load)
  - Register to memory (store)

- Load/store basic syntax

  \[
  \text{LDR} \quad r0, [r1] @ \quad \text{load into register from memory} \\
  \text{MOV} \quad r0, r1 @ \quad r0 := r1 \\
  \text{STR} \quad r0, [r1] @ \quad \text{Store from register into memory}
  \]

  Must be a register that contains a valid memory location

  \( r0 = \*r1 \quad (\text{In C}) \)
Base Register Addressing Mode

- The memory location to be accessed is held in a base register
  - STR r0, [r1] @Store contents of r0 to location pointed to @ by contents of r1.
  - LDR r2, [r1] @Load r2 with contents of memory location @pointed to by contents of r1.

```
Base Register
0x200
```

```
Source Register for STR
r0
0x5
```

```
Memory
0x200
0x201
0x202
0x203
```

```
Destination Register for LDR
r2
0x5
```

```
0x00 00 00 05
```

This is really 0x00 00 00 05
Assume Little Endian

Least significant byte goes into the lowest memory address
LDR r2, [r1]
The contents of which of the following changes when the above instruction is executed?

A. r1
B. r1 and memory
C. r2
D. r2 and memory
E. r1, r2 and memory
Data Transfer: Memory to Register

STR r2, [r1]

The contents of which of the following changes when the above instruction is executed?

A. R1
B. memory
C. R2
D. r1, r2 and memory
Variations on load/store

**Load**
- `LDR r0, [r1]` @ Load one word (4 bytes) into register r0 from memory
- `LDRH r0, [r1]` @ Load one half word (2 bytes) (unsigned)
- `LDRSH r0, [r1]` @ Load 2 bytes (signed)
- `LDRB r0, [r1]` @ Load 1 byte (unsigned)
- `LDRSB r0, [r1]` @ Load 1 byte (signed)

**Store**
- `STR r0, [r1]` @ Store 4 bytes
- `STRH r0, [r1]` @ Store least significant 2 bytes of r0 into memory
- `STRB r0, [r1]` @ Store LSB (one byte)

No `STRSH` or `STRSB`, why? With load, destination (register) is larger than for data

Little Endian

Memory: 0x200 → 0x201, 0x800 → 0x801, 0x001

Little Endian

LDRH

8001

1111 1111

0000 0000

Ms bit
Data Transfer Instructions

In C:

```c
void foo (int *p){
    *p=10;
}
```

Which of the following instructions is needed to translate the above code to ARM?

A. LDR and MOV
B. STR and MOV
C. Only LDR
D. Only STR

Mapping:

```
foo:
    mov r1, #10
    str r1, [r0]
```

STR, #10, [r0] is incorrect syntax
Data Transfer Instructions

In C:

```c
void foo (int *p){
    int a=*p;
}
```

In ARM:

```
LDR r1, [r0] @ r1 = *ro
```