COS10004 Computer Systems
Assignment 2

Daniel Coady (102084174) – 12:30 Wednesday

14/10/2019

1 mov

1.1 Syntax

```
mov x, y
```

Where:

- x is the destination
- y is the value

1.2 Description

Used to move a value into a register. Note that values must have 24 consecutive zeroes in it's binary notation.

1.3 Example

```
mov r0, $3F0000 ; valid
mov r0, $003F00 ; valid
mov r0, $00003F ; valid
mov r0, $300F00 ; invalid
```

2 orr

2.1 Syntax

```
orr x, y
```

Where:

- x is value 1 and the destination
- y is value 2

2.2 Description

Performs a bitwise OR operation on x and y, storing the result in x.

2.3 Example

```
mov r0, $10 ; r0 has 0x10
orr r0, $01 ; r0 has 0x11
```

3 ldr

3.1 Syntax

```
ldr x, [y]
```

Where:

- x is the register to store the value in
- y is the location to get the value from

3.2 Description

A pseudo instruction for storing 32-bit values in memory.

3.3 Example

4 ldrd

4.1 Syntax

```
ldrd x, y, [z]
```

Where:

- x is the register for the least significant half of the value
- y is the register for the most significant half of the value
- z is the location to get the value from

4.2 Description

Allows for storing of a 64-bit value across 2 32-bit registers.

4.3 Example

```
ldrd r0, r1, [r2, #4]
```

5 str

5.1 Syntax

```
str x, [y]
```

Where:

- x is the value to store
- y is the location to store the value into

5.2 Description

Used to store values within registers.

5.3 Example

```
str r0, [r1, #4]
```

6 add

6.1 Syntax

```
add x, y, z
add y, z
```

Where:

- x is the destination for the result
- y is a register holding the first number to add
- z is the second number to add

6.2 Description

Adds two numbers together. If x is not specified, then y becomes the destination.

6.3 Example

```
add r0, r1, #1
add r0, #1
```

7 sub

7.1 Syntax

```
sub x, y, z
sub y, z
```

Where:

- x is the destination for the result
- y is a register holding the first number to subtract
- z is the second number to subtract

7.2 Description

Subtracts z from y. If x is not specified, then y becomes the destination.

7.3 Example

8 mul

8.1 Syntax

```
mul x, y, z
mul y, z
```

Where:

- x is the destination for the result
- y is a register holding the first number to multiply
- z is the second number to multiply

8.2 Description

Multiplies y and z. If x is not specified, then y becomes the destination.

8.3 Example

```
mul r0, r1, #2
mul r0, #2
```

9 b

9.1 Syntax

```
bx y
```

Where:

- x is the condition for branching
- y is the label to branch to

9.2 Description

Branch instruction that allows for jumping to labels in code.

9.3 Example

```
loop:
  ; do some stuff
b loop
```

10 push

10.1 Syntax

```
push x
push (x, y, z, ...)
```

Where:

- x, y, z, ... is the value to push onto the stack

10.2 Description

Allows for pushing of values from registers onto the stack.

10.3 Example

```
push #1
push (#1, #2, #3)
```

11 pop

11.1 Syntax

```
pop x
pop (x, y, z, ...)
```

Where:

- x, y, z, ... is the register to store the value popped off the stack

11.2 Description

Allows for popping of values off the stack into registers.

11.3 Example

```
pop r0
pop (r0, r1, r2)
```

12 lsl

12.1 Syntax

```
lsl x, y
```

Where:

- x is the register holding the value to shift
- y is the amount to shift the value by in decimal

12.2 Description

Logical shift left of a binary value.

12.3 Example

```
mov r1, #1
lsl r1, #24
```

13 lsr

13.1 Syntax

```
lsr x, y
```

Where:

- x is the register holding the value to shift
- y is the amount to shift the value by in decimal

13.2 Description

Logical shift right of a binary value.

13.3 Example

```
mov r1, $0000FF
lsr r1, #10
```

14 cmp

14.1 Syntax

```
cmp x, y
```

Where:

- x is the first value to compare
- y is the second value to compare

14.2 Description

Compares two values to allow for conditional checks.

14.3 Example

```
cmp r0, #1
```