Heather Warman and David Merrick
April 29, 2014
CS 472 HW 4

(3.8) What are the relative advantages and disadvantages of general-purpose registers compared to separate address and data registers?

The Condition Code Register (CCR) stores the status after instructions. This register can indicate that the operation yielded a zero answer, a negative answer, an overflow, or a carry. Having general-purpose registers makes programming easier and more flexible, as data and addresses can be stored in any register. However, it means that the programmer has to manually update the CCR.

(3.9) What is a misaligned operand? Why are misaligned operands such a problem in programming?

Operand alignment means that the operand length is divisible by 1, 2, 4, or 8. This restricts it to be 1, 2, 4, or 8 bytes. If an operand falls outside of these parameters, it is said to be misaligned. If this is the case, it adds overhead for the CPU to access it. To access the operand, the CPU has to load multiple chunks of data, shift out the unnecessary bytes, and combine them. This results in a performance hit. Reference: http://www.songho.ca/misc/alignment/dataalign.html.

(3.24) What is the meaning of each of the P, U, B, W, and L bits in the encoding of an ARM memory reference instruction?

P bit: Specifies pre/post adjust. Setting this to 1 means to adjust the pointer before using. Works in conjunction with the W bit to determine how indexing is implemented.

U bit: Specifies pointer direction. Setting this to 0 means to decrement the pointer. The U bit defines whether the effective address should be calculated by adding or subtracting the offset.

B bit: Specifies whether the address is a byte or a word. Setting this to 0 means that it is a word address.

W bit: Specifies pointer writeback. Setting this to 1 means to update the pointer after use.

L bit: Specifies whether to load data from or store data to memory. Setting this to 0 means to store data in memory.

(3.26) What is the effect of LDR r0, [r5,r6, LSL r2]?

If R5 points to array X, and R6 contains index i, and R6 is scaled by a factor specified in R2, this instruction loads element located at the index of an array into register r0. Reference: book, pg 188.

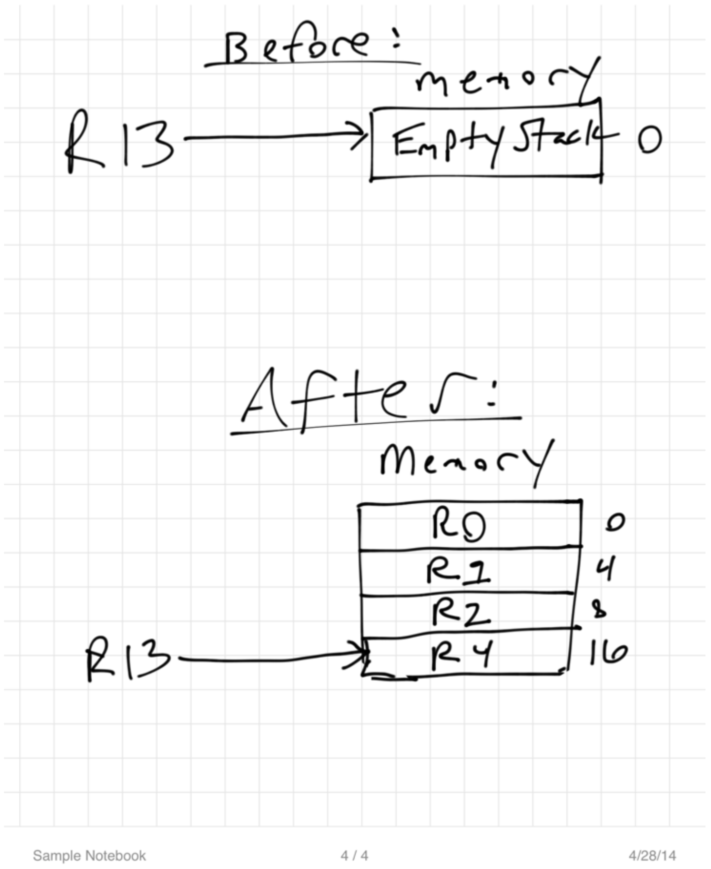
(3.30) What is the meaning of sign-extension in the context of copying data from one location to another?

Sign-extension means to increase the number of bits in a binary number while preserving its sign (positive or negative). For example, to sign-extend the value +4 in 8-bit binary to 10-bit binary, you would simply pad 0s on the left like so: 0000 0100 => 00 0000 0100. To sign extend the value -4 in twos complement form from 8 to 10 bits, you would pad 1s. 1111 1100 => 11 1111 1100 Source: http://en.wikipedia.org/wiki/Sign_extension. In the context of copying data from one location to another, sometimes data is stored in smaller words than the size of the destination register. If this is the case, the data will need to be padded (i.e. sign-extended) to preserve its value.

(3.33) Most RISC processors do not include a block move instruction. What are the advantages and disadvantages of the ARMs LDM and STM instructions?

STM is a block move-to-memory instruction. LDM is a block load-from-memory instruction. These are used to copy groups of registers to and from memory. ARM always stores the lowest register at the lowest memory address. Adding the IA suffix to STM indicates that the register containing the memory pointer should be incremented after the operation. ARMs block move instructions are versatile because they support the four possible varieties of stacks: full descending, full ascending, empty descending, and empty ascending. In summary, ARMs block move instructions have the advantage of being versatile in their ability to copy groups of registers to and from memory, but this versatility comes with the disadvantage of being somewhat complex to implement initially.

(3.34) What is the effect of executing STMIB r13!, r0-r2,r4? Draw a picture of the state of the stack pointed at by r13 before and after this operation.



This instruction will store R0-R2 and R4 on the stack. The exclamation point indicates auto-indexing, so the stack pointer will be updated afterwards.

(3.36) Without using the ARMs multiplication instruction, write one or more instructions (using ADD, SUB, and shifting) to multiply by the following integers. a. 33

MOV r1, #2 ; number to multiply

ADD r4,r1,r1, LSL #5 ; multiply by $2^5 + 1(33)$

b. 1025

MOV r1, #2 ; number to multiply

ADD r4,r1,r1, LSL #10 ; multiply by $2^{10} + 1(1025)$

c. 4095

MOV r1, #2 ; number to multiply

RSB r4,r1, LSL #12 ; multiply by $2^{12} - 1(4095)$

(3.44) What does the following code do?

TEQ r0,#0 ; Test the value of r0 against 0. If the values are equal, condition code flags will be updated but the result will not be placed in any register.

RSBMI r0,r0,#0 ; RSB means to reverse subtract. The MI suffix is a condition code meaning negative. So, if the result of the previous instruction was negative, subtract r0 from 0, and store the result in r0.

So, in summary, this code finds the absolute value of the number stored in r0 and stores it in r0.

(3.48) What, in the contest of assembly language, is a pseudo-operation?

A pseudo-operation is an instruction available to a programmer that is not actually part of the ARM ISA. During the build process, the assembler automatically translates this instruction into a set of other instructions that perform analogous work. Examples of pseudo-operations available to ARM programmers include the LDR and STR instructions.

(3.54) Explain what this fragment of code does instruction by instruction and what purpose it achieves (assuming that register r0 is the register of interest). Note that the data in r0 must not be 0 on entry.

MOV r1,#0 ; place the literal value 0 in r1 loop MOVS r0, r0, LSL #1 ; The S suffix means to update the condition code flags on the result of the operation. This instruction shifts r0 1 bit to the left. By setting the S flag, this will generate a carry in the event of a leading 1.

ADDCC r1,r1,#1 ; The CC suffix means that if the unsigned result is less than one, increment r1 by 1.

BCC loop ; branch if CC (carry clear); Return to the start of the loop if the result of the previous operation did not result in a carry.

This code counts the number of leading 0s in r0 and stores the count in r1.

(3.60) A computer has three eight-element vectors in memory, Va, Vb, and Vc. Each element of a vector is a 32-bit word. Write the code to calculate all elements of Vc.

```
AREA calculate_vector, CODE, READWRITE
; Setup the memory addresses for the vectors
Va DCD 8,7,6,5,4,3,2,1
Vb DCD 1,2,3,4,5,6,7,8
; Set Vc to all ones (Actual values are set during runtime)
; This should result in all 4s in Vc
Vc DCD 1,1,1,1,1,1,1,1
; Point registers at vector memory addresses
ADR r1, Va ; point r1 to Va memory address
ADR r2, Vb ; point r2 to Vb memory address
ADR r3, Vc ; point r3 to Vc memory address
; Setup loop counter, init to 0
MOV r8, #0
vectorloop LDR r4, [r1], #4 ; load a word from Va, store in r4
LDR r5, [r2], #4 ; load a word from Vb, store in r5
CMP r8, #8 ; while loop counter  8, do calculations
BEQ finished ; branch when done
ADD r8, #1 ; increment loop counter by 1
; add r4 and r5, store result in r6
ADD r6, r4, r5
; divide r6 by 2
; (1 bit-shift right divides by 2)
MOV r6, r6, ASR #1 ; (ASR: Arithmetic Shift Right)
; save final value to Vc in memory
STR r6, [r3], #4 ; Store a word from r6 to memory address in r3 (Vc)
b vectorloop
```

finished b finished

; Throw some random values in for the vectors

END

(3.61) Register r15 is the program counter. You can use it with certain instructions such as a MOV (e.g., MOV pc,r14). However, r15 cannot be used in conjunction with most data processing instructions. Why?

R15 is the program counter. This means that it contains the address in memory of the next line of code to execute. If R15 is used as a pointer register, it will execute the next instruction based on whatever offset it contains. Data processing instructions store the result of the operation in the destination register. If R15 is used as the destination register, the result of the data processing instruction would be stored there, and the CPU would next execute an instruction with an offset equal to that result. This would result in erratic code execution, making it very difficult if not impossible to retrieve the result of the data processing instruction without knowing the value of that result beforehand.