ASM Dictionary

Write a manual covering every ASM command we have used this semester.
You must include:
• The command syntax
• What it does (incl. what registers it changes)
• Example code
• Special instructions on using the operation (e.g. must use an even register)
• 3 or four lines of text will be sufficient for most ASM operations.

Assignment

MOV

MOV{S}{cond} Rd, Operand2
MOV{cond} Rd, #imm16
The Mov instruction copies the value of Operand2 into Rd.
Example
MOV r1, #1 ; loads register 1 with the value 1

LDR

LDR{cond}{.W} Rt, =expr
LDR{cond}{.W} Rt, =label_expr
Load a register with either a 32 bit immediate value or an address.
Used to MOV an illegal number (without 8 consecutive 0s in the binary representation)
Tries to perform mov operation and converts the immediate value to other things to fudge a working solution.
Example
Ldr r0, [r1] ;loads register 0 with the value pointed to by r1

STR

STR{type}T{cond} Rt, [Rn {, #offset}]
STR{type}T{cond} Rt, [Rn] {, #offset}
Stores byte, halfword or word to a register by writing a value from register to a memory location.
Example
Str r1, [r0,#16] ;write value r1 into the memory location, 16 bytes after the value in r0

LSL

LSL{S}{cond} Rd, Rm, Rs
LSL{S}{cond} Rd, Rm, #sh
Provides the value of a register multiplied by a power of two, inserting zeros into the vacated bit positions.
Logical left shift, Doubling a number each shift.
Example
Lsl r1, #21 ;doubles register 1's value by 21 times r1^21

ORR
ORR{S}{cond} Rd, Rn, Operand2
Adds numbers together without overflow or carry operations
Construct any 32 bit number by combining parts of the number
Performs bitwise OR operations on the values in Rn and Operand2
Example
Orr r1, $21 ;or statement adding 21 to register 1

Arithmetic
ADD
ADD{S}{cond} {Rd}, Rn, Operand2
Adds the values in Rn and Operand2 or imm12
Add values together without carry
Includes overflow and carry operations
Example
Add r0,r0, #2 ;r0+=2

SUB
SUB{S}{cond} {Rd}, Rn, Operand2
SUB{cond} {Rd}, Rn, #imm12
Subtracts the value of Operand2 or imm12 from the value of Rn.
Subtracts values from each other without carry.
Example
Sub r2, #1 ;subtract 1 from register 2

MUL
MUL{S}{cond} {Rd}, Rn, Rm
Multiply with signed or unsigned 32 bit operands, giving the least significant 32 bits of the result.
Multiples the values from Rn and Rm, and places the least significant 32 bits of the result in Rd.
Example
Mul r2, r1, r3 ;multiples r3 and r1 and stores result in r2

Labels and branch
Labels
label:
Name for function or variable
Labels must be unique
PC relative expression
Example
LOOP:
CONSTANT = $3F000000

B
B{cond}{.W} label
Causes a branch to label
Branches from one function to either the same one creating a loop or breaking off to another.
Example
Loop:
B Loop

Include
Include “filename”
Provides a way to include supporting files at specified points in your source program
Allows you to add different files contents to a file so that you can use the functions
Example
Include “GPIO.asm”

Commenting
;
Allows you to make comments about the code
Example
;activates the GPIO pin

Registers
Each raspberry pi has registers which can store 32 bytes, as either an address or a value. The raspberry Pi has 16 registers which can be accessed in ASM with r followed by a number between r0-r15.

Application Binary Interface
Standard way of using ARM registers
R0-r3 used for function arguments and return values
R4-r12 promised not to be altered by functions
Lr and sp used for stack management
Pc is the next instruction used to exit a function call

LR link register
Contains the address of the next instruction after a function call.
Used to tell the code what to run after a function finishes

PC program counter
Current address of code to be run is stored here
Setting this to the value in lr makes the program resume after a function has finished

Selection

CMP

CMP{cond} Rn, Operand2
Compare the value in a register with operand2. They update the condition flags on the result, but do not place the result in any register.
Subtracts the value of Operand2 from the value in Rn
Subtracts 2nd value from first and sets flags
Loads the Application Program Status Register(APSR)
Sets flag to identify the cmp of the result
Example
Cmp r1,r2 ;compares the value stored in r1 and r2

TST

TST{cond} Rn, Operand2
Tests the value in a register against Operand2. It updates the condition flags on the result but does not place the result in any register
Example
TST r0, #0x3F8

APSR

APSR flags
Values are set when using a cmp or tst from the result.

N - result was negative
Z - result was zero
C - set the carry bit
V result caused overflow

Condition Code

Cond in glossary.
Used like conditional statements in programming to branch of if certain conditions are met.
Informs you of the result of a compare by storing a value to tell if the condition is correct
Used like a if statement
Example
Beq loop ;branch if compare is equal

Suffix	Flags	Meaning
EQ	Z set	Equal
NE	Z clear	Not Equal
CS or HS	C set	Higher or same (unsigned >=)
CC or LO	C clear	Lower (unsigned <)

MI	N set	Negative
PL	N clear	Positive or zero
VS	V set	Overflow
VC	V clear	No overflow
HI	C set and Z clear	Higher (unsigned >)
LS	C clear or Z set	Lower or same (unsigned <=)
GE	N and V the same	Signed >=
LT	N and V differ	Signed <
GT	Z clear, N and V the same	Signed >
LE	Z set, N and V differ	Signed <=

Stack

Push

PUSH{cond} reglist
Push register onto a full descending stack
Adds value to the top of the stack
Example
Push{r2, r3}; puts r2 and r3 on the stack

Pop

POP{cond} reglist
Pop register off a full descending stack
Removes item from the top of the stack
Example
Pop{r2, r3} ;removes r2 and r3 from the stack

Bx

BX{cond} Rm
Branch and exchange instruction set
Causes a branch to the address contained in Rm and exchanges the instruction set.
Example
Bx lr; ;Branch to lr without updating lr

BL

BL{cond}{.W} label
Branch with link

Causes a branch to label, and copies the address of the next instruction into LR the link register.
Example
BL loop ;branches to loop label

Aliases/Variables

.req

Label .req Rt
Defines a register alias
Example
NAME .req r0

.unreq

.unreq Label
Unlinks a register alias which was defined with req
Example
.unreq NAME

Symbols

$ Hex value

# Decimal value

Glossary of Terms

Imm16: is any value in the range of 0-65535
.W: is an optional instruction width specifier
Expr: evaluates to a numeric value
Label_expr: is a PC-relative or external expression of an address in the form of a label plus or minus a numeric value
Type: can be B for Byte, H for halfword and omitted for word.
Cond: is an optional condition code.
Offset: is an offset, if omitted the address is the value of Rn.
S: is an optional suffix. If S is specified, the condition flags are updated on the result of the operation
Operand2: is a flexible second operand
Rd: is the destination register
Rm: is the register holding the first operand. This operand is shifted left or is a register containing an address to branch to.
Rn: is the register holding the first operand or is the register on which the memory address is based
Rs: is a register holding a shift value in Rm. Only the least significant byte is used.
Rt: is the register to load or store

Sh: is a constant shift. The range of values permitted is 0-31.
Label: PC-relative expression. Used like a variable
Reglist: is a non empty list of registers, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range.
Filename: name of a file includes the file type

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473m/DUI0473M_armasm_user_guide.pdf