ARM Assembler

Basics

Contents

1 About ARM

- RISC Architecture (Reduced Instruction Set Computing)
- Operates only on registers, NOT on memory
  - only load/ store instructions can access memory
  - incrementing an integer in memory requires load, increment and store operations
- operations can be executed more quickly then in intel arch
- Two modes: ARM and Thumb mode
- Most instructions can be used for condidional execution
- Endianness depends on ARM version, since ARM v3 even Bi-endian

2 Assembly

Official documentation: infocenter.arm.com/help/index.jsp

- Can be assembled using **as** from GNU Binutils
- .s extension
- **as** produces object files (.o) that have to be linked
- Can be linked using **ld** from GNU Binutils

```
$ as program.s -o program.o
$ ld program.o -o program
```

3 Data types

- Signed and unsigned data types
- Operations have datatype specific extensions

Data type	Size	Extension
Byte	8 Bits	-b
Signed Byte	8 Bits	-sb
Half Word	16 Bits	-h
Signed Half Word	16 Bits	-sh
Word	32 Bits	none
Signed Word	32 Bits	??

Load and store example:

```
ldr = load word
ldrh    = load unsigned half word
ldrsh   = load signed halfword
...

str = store word
strh    = store unsigned half word
strsh   = store signed half word
...
```

4 Registers

- Amount of registers depends on ARM version
- Accordingf to ARM reference manual there are 30 32 Bit universal purpose registers

- First 16 registers are accessible in user mode, additional registers only accessible in privileged software execution (Except in ARMv6-M and ARMv7-M)
- First 16 registers can be split in general purpose and special purpose registers

#	Alias	Purpose
R0	-	General Purpose
R1	-	General Purpose
R2	-	General Purpose
R3	-	General Purpose
R4	-	General Purpose
R5	-	General Purpose
R6	-	General Purpose
R7	-	Holds Syscall number
R8	-	General Purpose
R9	-	General Purpose
R10	-	General Purpose
R11	FP	Frame Pointer
R12	IP	Intra Procedural Call
R13	SP	Stack Pointer
R14	LR	Link Register
R15	PC	Program Counter
CPSR	-	Current Program Status Register

4.1 R0 - R12

- Can be used during common operations to store temp values, memory addresses (pointers), etc.
- R0 can be used as accumulator
- R7 stores syscall number
- R11, stack frame pointer, helps keep track of stack boundaries
- ARM function calling convention uses R0 - R3 to pass the first four function arguments

4.2 R13: SP (Stack Pointer)

- Points to the top of the stack
- Stack memory is allocated by subtracting (in bytes) from SP
- e. g. to allocate 32 Bit of the stack we subtract 4 from SP

4.3 R14: LR (Link Register)

- Stores return address when a function is called (next instruction after function call)
- Allows the program to resume in "parent function"

4.4 R15: PC (Program Counter)

- Is automatically incremented by the size of the instruction executed
  - ARM state: 4 bytes
  - Thumb state: 2 bytes
- When branching, PC holds the destination address
- During execution of an instruction PC holds the instructions address + 8 (two ARM instructions) in ARM mode or current instruction address + 4 (two Thumb instructions) in Thumb mode
- ***This is different from x86 where PC always holds address of next instruction to be executed!***

4.5 CPSR: Current Program Status Register

- Represents program status as flags
- Used for conditionals

Flag	Description
N (Negative)	Enabled if result of instruction yields a negative number.
Z (Zero)	Enabled if result of instruction yields a zero value.
C (Carry)	Enabled if result of instruction yields a value that requires a 33rd bit to be fully represented
V (Overflow)	Enabled if result of instruction yields a value that cannot be represented in 32 bit two's complement
E (Endian Bit)	ARM can operate in big or little endian mode. 0 represents little endian, 1 represents big endian.
T (Thumb Bit)	Enabled if in Thumb mode
M (Mode bits)	These bits represent the current privilege mode.
J (Jazelle)	Third execution state that allows some ARM processors to execute Java bytecode in hardware.

5 ARM Instructions

TODO: Thumb instructions.

5.1 Instruction Syntax

```
MNEMNOTIC{S}{sondition} {Rd}, Operand1, Operand2
```

MNEMNOTIC	Short name (mnemnotic) of instruction.
{S} An o	ptional suffix. If S is specified the conditional flags are updated on the result of the operation.
{condition}	Condition that is needed to be met for the instruction to be executed.
{Rd}	Register (destination} for storeing the result of the instruction.=
Operand1	First operand. Either register or immediate value.
Operand2	Second (flexible) operand. Can be immediate value (number) or register with an optional shift.

5.2 Most common ARM mode instructions

Instruction	Description
MOV	Move Data
MVN	Move and negate
ADD	Addition
SUB	Subtraction
MUL	Multiplication
LSL	Logical shift left
LSR	Logical shift right
ASR	Arithmetic shift right
ROR	Rotate right
CMP	Compare
AND	Bitwise AND
ORR	Bitwise OR
EOR	Bitwise XOR
LDR	Load
STR	Store
LDM	Load multiple
STM	Store multiple
PUSH	Push on Stack
POP	Pop from stack
B	Branch
BL	Branch with link
BX	Branch and exchange
BLX	Branch with link and exchange
SWI / SVC	System call