ADVANCED PROCESSOR ARCHITECTURES AND MEMORY ORGANISATION –

Lesson-13: ARM Instruction Set

Features

- Two Instruction Sets— 16-bit Thumb and 32-bit ARM mode instructions
- Operations on 8-bit or 16-bit or 32-bit data types
- Data alignment in memory: Two byte words in Thumb set and Four in 32-bit ARM mode

ARM7 instruction set: Data Transfer Instructions

- Register-load a byte (*LDRB*).
- Register- byte store (*STRB*).
- Register Half Word store (*STRH*). [A word in ARM is of 32 bits].
- Register-load Half Word as such or signed (*LDRH* or *LDRSH*).

ARM7 instruction set: Data Transfer Instructions

- Instructions for transfer between the register-memories. The memory address is as per a register used as index or index-relative or post auto-index addressing mode.
- Register-load a word (*LDR*).
- Register-word stores a word (*STR*).
- Set a memory address into a register (*ADR*). Address is of 12 bits. [Alternative for 16 bits address setting in a register is using any register or *r15* in an arithmetic operation].

Word transfer between registers:

- Move (*MOV*).
- Move after Negating (*MVR*).

Load or move or store instruction conditionally implementation

- Conditions─ signed number *LT*, *GT* , *LE*, EQ, *NE* (not equal), *VS* (overflow), *VC* (no overflow), *GE*
- Conditions─ unsigned number *HI* (higher), *LS* (lower), *PL* (plus, nor Negative), *MI* (minus), *CC* (carry bit reset), and *CS* (carry bit set).
- Example: *MOV**LT** r3*, #10.
- Immediate operand 10 to *r3* provided a previous instruction for comparison showed the first source as less than the second.

Bit Transfer or Manipulation Instructions

- Register- bits Logical Left Shift (*LSL*).
- Register- bits Logical Left arithmetic Shift (*ASL*).
- Register- bits Logical Right Shift (*LSR*).
- Register- bits Logical Right arithmetic Shift (*ASR*).
- Register- bits Rotate Right (*ROR*).
- Register bits Rotate Right with carry also extended for rotating (*RRX*).

Arithmetical Instructions

- Three operands from the registers.
- One source may however, be by immediate operand addressing in addition and subtraction .
- Add without carry two words and the result is in the third operand (*ADD*).
- Add with carry two words and the result is in the third operand (*ADC*).
- Subtract without carry two words and the result is in the third operand (*SUB*). [Carry bit used as borrow.]
- Subtract with carry two words and the result is in the third operand (*SBC*).

Arithmetical Instructions

- Subtract reverse (second source with the first) without carry two words and the result is in the third operand (*RSB*). [Carry bit used as borrow.]
- Subtract reverse with carry two words and the result is in the third operand (*RSC*).
- Multiply two different registers and the result is in the destined register (*MUL*).
- Multiply two source registers and add the result with the third source register and accumulate the new result in a destined register. (*MLA*).

Logic Instructions

- Bit wise OR two words and the result is in the third operand. (*ORR*).
- Bit wise AND two words and the result is in the third operand. (*AND*).
- Bit wise Exclusive OR two words and the result is in the third operand. (*EOR*).
- Clear a Bit (*BIC*). [There is one source for the bits; a second source for the mask and the result is at the third operand.]

Arithmetical or logical instruction conditional implementation

- Example, SUB***GE*** *r1*, *r3*, *r5*. The operand from r3 is subtracted from r5 if the GE condition resulted earlier (N and V status bits equal on comparison of two signed numbers).
- Conditions can be the results of a *comparison* or *test*

Compare and Test Instructions

- *The* result destines to *CPSR, which stores the four* condition bits, N, V, C, and Z.
- Bit wise Test two words (*TST*).
- Bit wise Negated Test between two words (*TEQ*).
- Compare two words and the result is at the *CPSR* condition bits (*CMP*).
- Compare two negative words and the result is at the *CPSR* condition bits (*CMN*).

Program-Flow Control Instructions

- Branching (B) or Branch conditional operations.
- Branch to an address relative to *PC* word in *r15* (*B*) '*B* #1A8' means add in *PC* 1A8 and change the program flow.
- '*BGE* #100' means that if a *GE* condition resulted on a compare 0 test, add in *PC* 1A8.
- Similar instructions for different conditions of the processor status flags

Software Interrupt instruction

- SWI has 8-bit opcode and remaining bits are not used by processor
- Give single vector address of the ISR for SWI.
- Remaining bits in SWI backtracked by programmer to compute ISR and ISR parameter pointers
- This unique feature permits handling large number of SWIs required in the OS and application functions or threads or tasks

Summary

We learnt

- ARM Architecture Instruction set
- Byte, half word and 32-bit data types
- 16-bit Thumb and 32-bit ARM mode instructions
- world length Data transfer, arithmetic, logic instructions
- Program flow control instructions
- Software interrupt instruction

End of Lesson 13 of Chapter 2