SPRING 2016

CMPE 364

Microprocessor Based Design

Dr. Mohamed Al-Meer

Subroutines

ARM Architecture

Lecture Expectations

- Expected to achieve:
  - Know about how to call subroutines and return
  - Return with branch and exchange
  - Preserving link register
  - AAPCS standard
  - Parameter passing and return values
  - Other special function registers
  - Register preservations (across Calls)

Introduction

- Subroutine is a subprogram within your program which accomplishes a specific task.
- Advantages:
  - Improve readability. **How? Why?**
  - Simplify design. **How? Why?**
  - Simplify program testing. **How? Why?**
  - Increase modularity. **How? Why?**
  - Easy program modification & addition. **How? Why?**
  - Code reuse. **How? Why?**
  - Selective Users Contribution. **How? Why?**

Introduction

- See next code

```
BL    GET_INPUT
MOV R1, R0
BL    GET_INPUT
ADD  R0, R0, R1
BL    DISPLAY_RESULT
```

- Takes 2 number from user. Add numbers. Display Sum.
- Each task is isolated in one subroutine

Calling & Returning

- **BL** instruction will be used (Branch and Link)
- Return address is saved automatically in **LR** register (**R14**)
  - Address of instruction after BL will be saved in LR register
  - Control transfers to label
  - At end of subroutine program copies LR to PC

Conditional Branch and Link

- BL can be conditional or unconditional
- What applies to B instruction applies to BL instruction
- Condition part of mnemonic directly follows BL or B
- See next Example when branch will be taken only if ZF=1
  - **BEQ DISPLAY_RESULT**
  - **BLEQ DISPLAY_RESULT**

Return from Branch

- To return from subroutine you can use:
  - **MOV PC, LR**
- Some development tools will generate an a warning. But it prefers to use the next one:
  - **BX rm**
  - where rm contains the branch target address
- Least significant bit of rm is a control flag and has a meaning.
  - If =0 then ARM mode is selected
  - If = 1 then Thumb mode is selected
- Then this bit will be 0 and ARM processor will execute the branch

Preserving Link Register

- **If a subroutine calls another subroutine?**, LR register must be preserved.
- Need to push LR to stack at start of routine and pop it at end of the routine before the return.

```
STMFD     SP!, {LR}
          --
          BL          ANOTHER ROUTINE
          --
LDMFD     SP!, {LR}
MOV       PC, LR      or    BX    LR
```

AAPCS!

- When writing Procedures:
  - **How data is passed in and out of routine?**
  - **How registers allocated inside the subroutine such that it does not overwrite others in the caller?**
- Is a **Procedure Call Standard for the ARM Architecture**.
  - Specifies set of rules for subroutine entry and exit
- See next table

AAPCS!

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Table 2, Core registers and AAPCS usage

AAPCS! Parameter Passing & Return

- According to AAPCS, **R0** through **R3** used to pass data and return values to/from subroutines
- Known as **a1 – a4**
- Before calling a subroutine parameters should be placed in argument register, to have data available for the routine
- Subroutines which return values are called functions
  - Used **R0** and **R1** as **return values**
  - So before return values should be placed in **a1** and **a2** (R0 and R1).

AAPCS! Parameter Passing & Return

- **EXAMPLE**

```
MOV a1, #588
MOV a2, #23
BL  DIVISION
```

- **EXAMPLE**

  Needed to CALL a routine which sorts an array. The address of array should be passed with the number of elements

```
LDR  v1, =0x40001000     Pseudo instruction
MOV  a1, 180             v1 pointer to array
BL   SORT                a1 length
```

AAPCS! Special Function Registers

- Parameter passing has been addressed
- **What about remaining registers in AAPCS?**
  - PC or r15 is program counter
  - LR or r14 is link register
  - SP or r13 points to TOS
- **IP** or **r12** or **I**ntra-**P**rocedure calls used as scratch register between routine and subroutine it calls.
  - Or used to store intermediate values between subroutine calls.

AAPCS! Special Function Registers

- **R9** is platform specific register. **Used as**:
  - May be used as static base (**sb**) in position *independent data model*
  - As *thread register* (**tr**) when thread local registers used
  - Case in virtual platform
  - Or **another variable register (v6)** ← Yes. Mostly used
    - **R9 as v6 will be used by us**.

AAPCS! Register Preservation across calls

- **R0 – R3** considered as scratch registers
  - **Means** routine uses them as local variables
  - Could be changed without effect on module who called this subroutines
  - If subroutines utilize variable registers or SP then they must be preserved.
    - Must have same value as before the call to the new subroutine.
  - Will use a technique called "**Register Spilling**" for preserving intended registers before after the call

AAPCS! Register Preservation across calls

- **EXAMPLE**
- **CALLER**

```
LDR  v1, =ARRAY
MOV a1, #10
MOV a2, #09
BL   SEARCH
--
--
```

- **SUBROUTINE**

```
SEARCH  STMFD    SP!, {v1}
        reg a1-a4 may be used
        without preserving values
        --
        --
        LDMFD    SP!, {v1}
        MOV      PC, LR
                 v1 restored^
```