

Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.
Electrical and Computer Engineering
Kettering University

Announcement

- There is no class on Friday (please check KU calendar).
- You are going to have quiz no.4 next Friday, Feb 12.
 - Covers homework exercise 5
 - Stack and subroutines

Lecture 15: Subroutines

Today's Topics

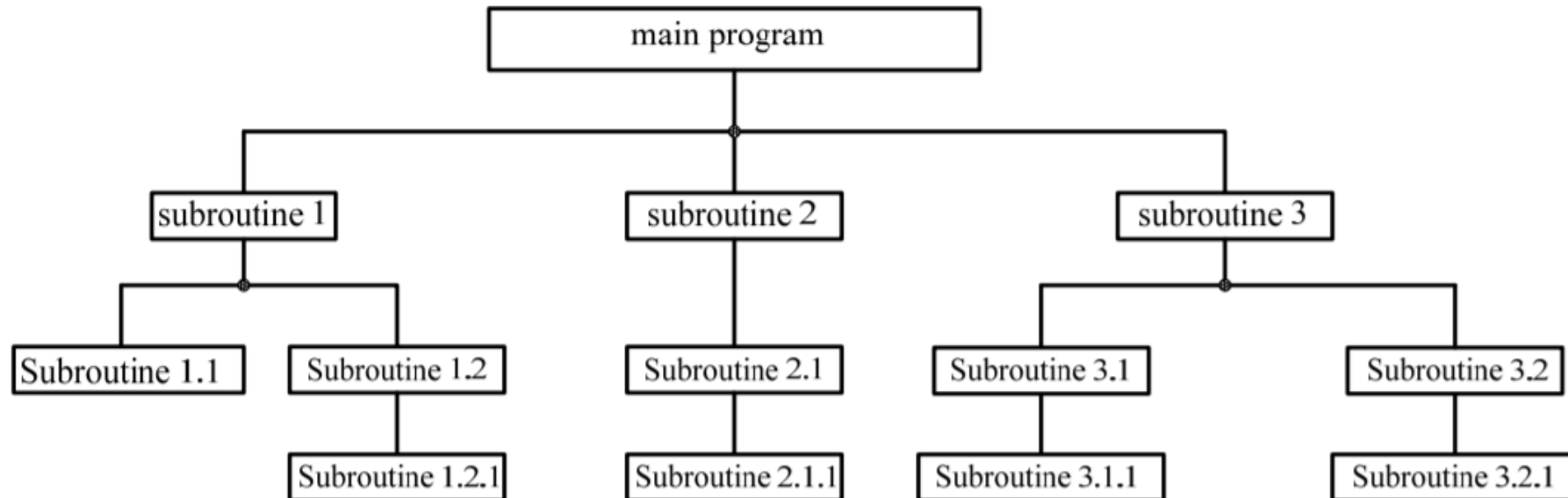
- What is subroutines?
- Learn how to call subroutines from an assembly program.
- Learn the properties of well-written subroutines.
- Learn what are pass-by-reference and pass-by-value.

Introduction

- Good program design is based on the concept of modularity
 - Modularity ➡ partitioning of a large program into subroutines
- Program can be divided into two sections
 - Main program
 - Subroutines
- Main program ➡ contains the logical structure of the algorithm
- Subroutines ➡ execute many of the details involved in the program

Subroutine Hierarchy

- Structure of a modular program can be visualized as shown
- Principles of program design involved in high-level languages can be applied to assembly language programs
 - Subroutine “objects” with calls and returns

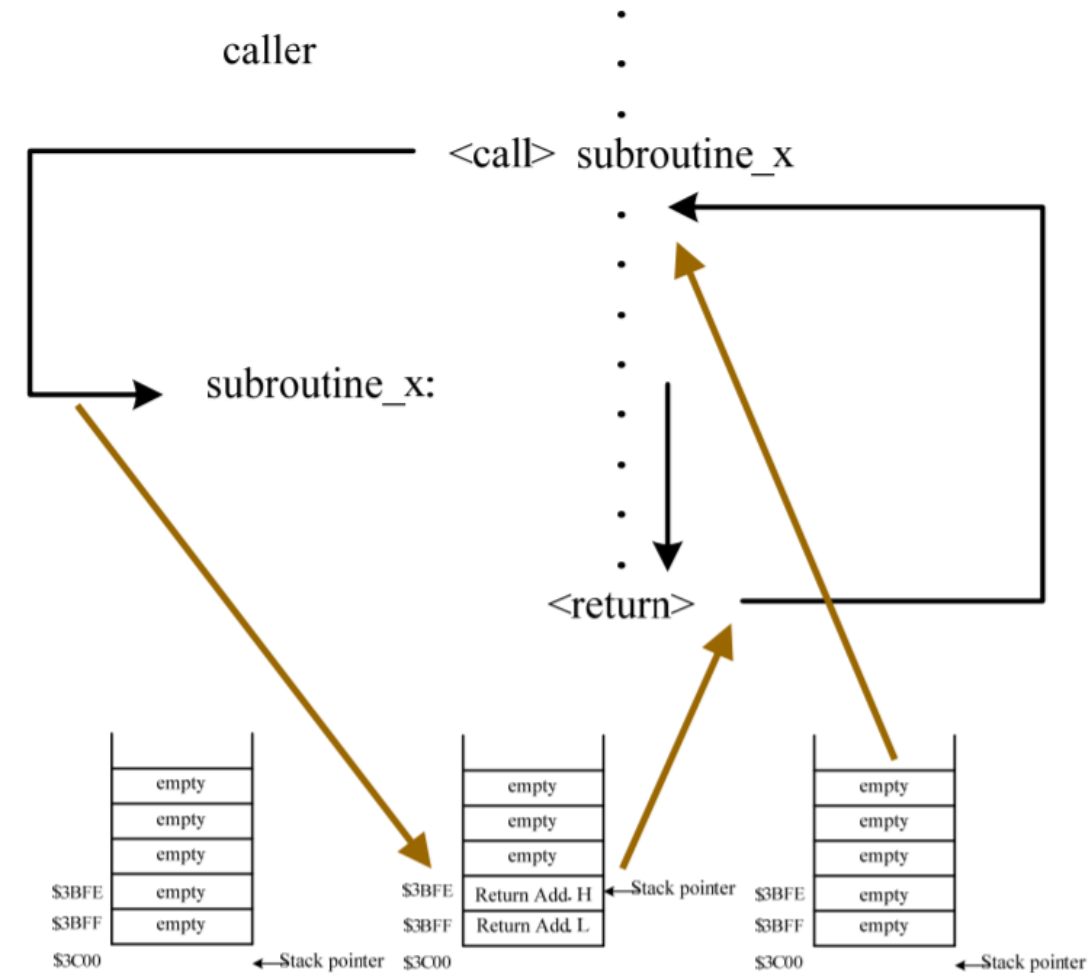


Subroutines

- Subroutine is a sequence of instructions that can be called from many different places in a program
 - One key issue in a subroutine call is to make sure that the program execution returns to the point immediately after the subroutine call when the subroutine completes its computation
- The address for returning to the point immediately after the subroutine completes its computation is known as the **return address**
 - This is normally achieved by saving and retrieving the return address in and from the **stack**

Subroutine Program Flow

- The subroutine call saves the return address, which is the address of the instruction immediately following the subroutine call instruction, in the stack system
- After completing the computation task, the subroutine will return to the instruction immediately following the instruction that makes the subroutine call
- This is achieved by executing a return instruction, which will retrieve the return address from the stack and transfer CPU control to it



Need Something for Subroutine Operation

Let's take a look at this code

- One of the main characteristics that makes a subroutine a subroutine is the ***ability to branch to (or return to) different addresses in the main program.***
- The need for this is demonstrated in the example here.

```

                ORG    $C000
                LDAA    #17      ; C000
                BRA     MagA     ; C002
Ret1            STAA    $1000    ; C004
                LDAA    #-1      ; C007
                BRA     MagA     ; C009
Ret2            STAA    $1001    ; C00B
                SWI           ; C00E

; Subroutine MagA
; compute magnitude of a single byte number
; input: byte in register A
; output: magnitude returned in register A

                ORG    $C200
MagA            TSTA
                BPL     return
                NEGA
Return         BRA     ?????
```

Subroutine Instructions

JSR and RTS

- **JSR** (Jump Sub Routine)
 - Pushes two-byte address of the next line of code on the **stack** first.
 - Jump/Branch to the subroutine.
 - Many addressing modes supported, but extended is often the most useful in Micros I.
- **RTS** (ReTurn from Subroutine)
 - Pulls two bytes off of the **stack** into the PC and jumps to that address

Example for Subroutines

```

      ORG    $C000
      LDS    #$3600          ; C000
      LDAA   #17              ; C003
Jsr1   JSR    MagA            ; C005
Ret1   STAA   $1000           ; C008
      LDAA   #-1              ; C00B
Jsr2   JSR    MagA            ; C00D
Ret2   STAA   $1001           ; C010
      SWI                      ; C013

; compute magnitude of a single byte number
; input: byte in register A
; output: magnitude returned in register A

MagA   TSTA
      BPL    return
      NEGA
Return RTS
```



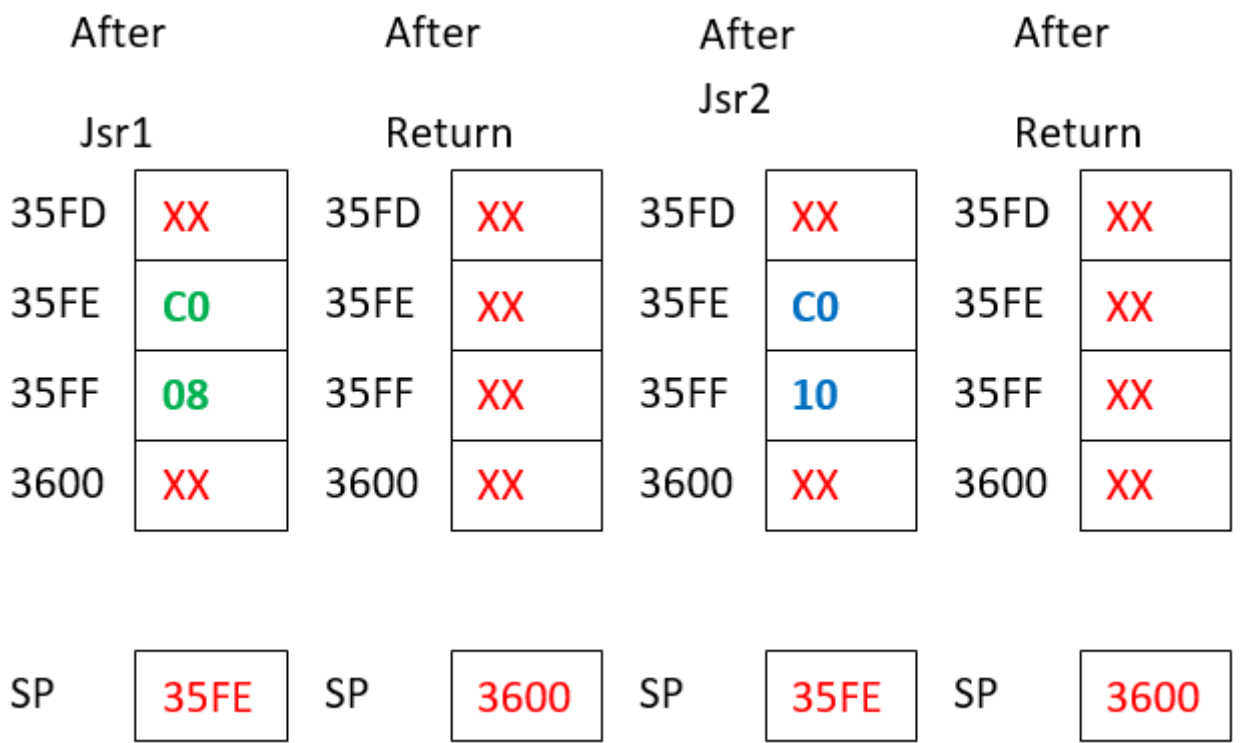
Example for Subroutines

```

      ORG    $C000
      LDS    #$3600          ; C000
      LDAA   #17              ; C003
Jsr1  JSR    MagA             ; C005
Ret1  STAA   $1000            ; C008
      LDAA   #-1              ; C00B
Jsr2  JSR    MagA             ; C00D
Ret2  STAA   $1001            ; C010
      SWI                      ; C013

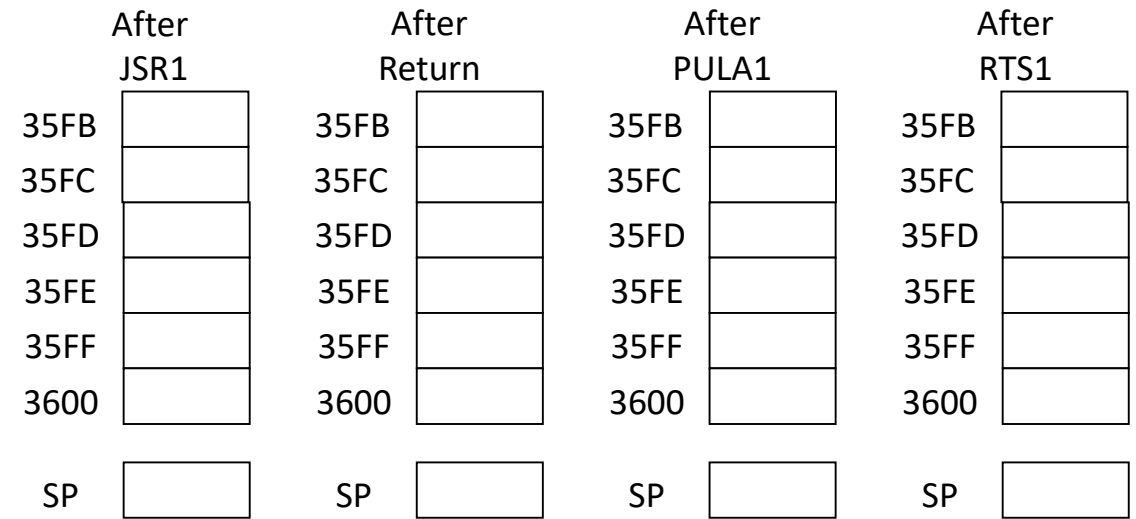
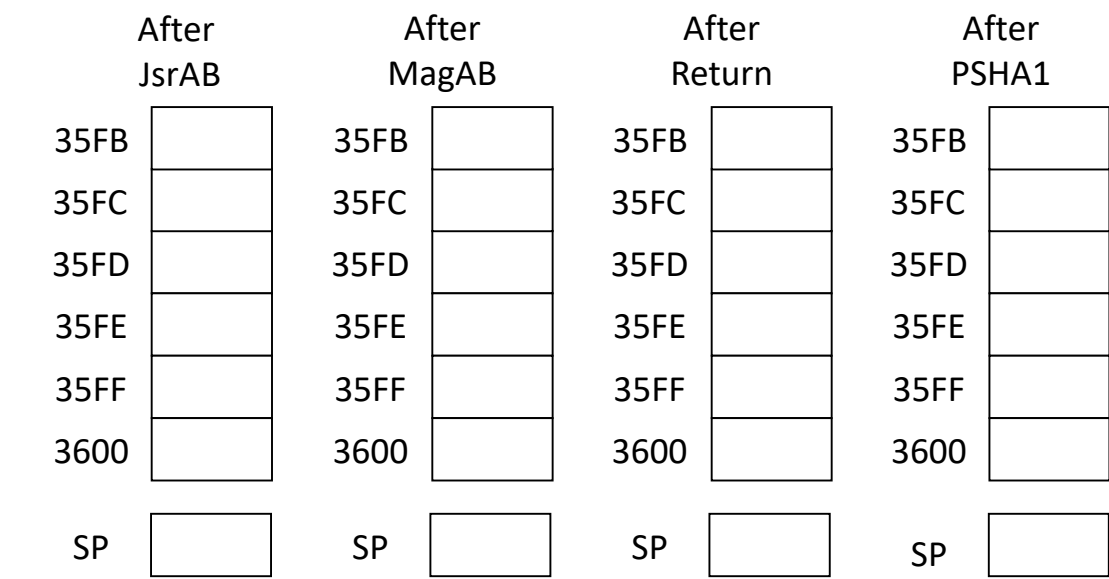
; compute magnitude of a single byte number
; input: byte in register A
; output: magnitude returned in register A

MagA  TSTA
      BPL    return
      NEGA
Return RTS
```



Nesting Subroutines

JsrAB	ORG	\$C000	
	LDS	#\$3600	; C000
	LDAA	#17	; C003
	LDAB	#-1	; C005
	JSR	MagAB	; C007
	SWI		; C00A
MagAB	JSR	MagA	; C00B
PSHA1	PSHA		; C00E
	TFR	B,A	; C00F
JSR1	JSR	MagA	; C011
	TFR	A,B	; C014
PULA1	PULA		; C016
RTS1	RTS		; C017
MagA	TSTA		; C018
	BPL	return	; C019
	NEGA		; C01B
Return	RTS		; C01C



Nesting Subroutines

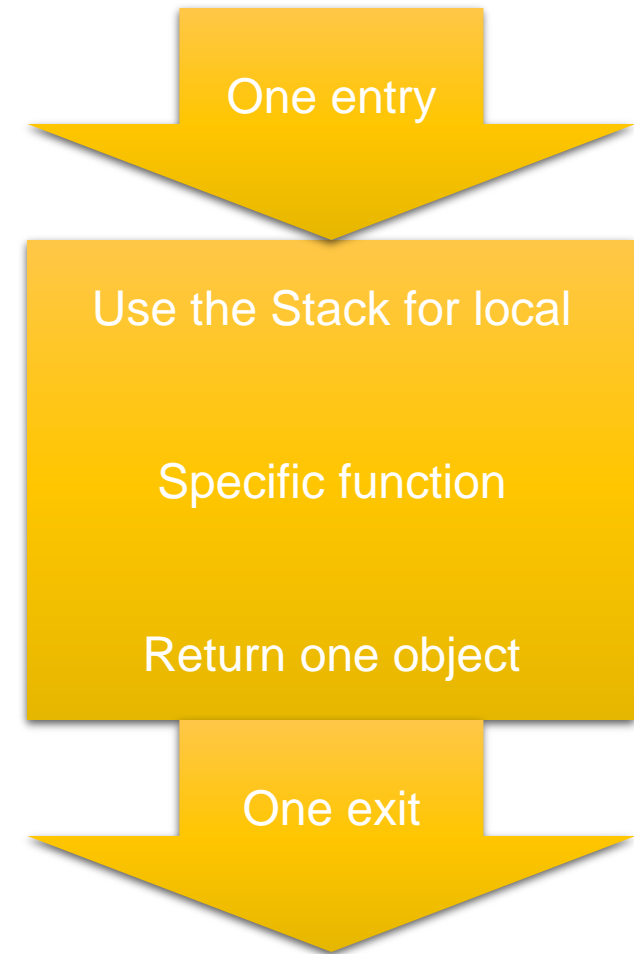
	ORG	\$C000	
	LDS	#\$3600	; C000
	LDAA	#17	; C003
	LDAB	#-1	; C005
JsrAB	JSR	MagAB	; C007
	SWI		; C00A
MagAB	JSR	MagA	; C00B
PSHA1	PSHA		; C00E
	TFR	B,A	; C00F
JSR1	JSR	MagA	; C011
	TFR	A,B	; C014
PULA1	PULA		; C016
RTS1	RTS		; C017
MagA	TSTA		; C018
	BPL	return	; C019
	NEGA		; C01B
Return	RTS		; C01C

After JsrAB	After MagAB	After Return	After PSHA1
35FB XX	35FB XX	35FB XX	35FB XX
35FC XX	35FC C0	35FC XX	35FC XX
35FD XX	35FD 0E	35FD XX	35FD 17
35FE C0	35FE C0	35FE C0	35FE C0
35FF 0A	35FF 0A	35FF 0A	35FF 0A
3600 XX	3600 XX	3600 XX	3600 XX
SP 35FE	SP 35FC	SP 35FE	SP 35FD

After JSR1	After Return	After PULA1	After RTS1
35FB C0	35FB XX	35FB XX	35FB XX
35FC 14	35FC XX	35FC XX	35FC XX
35FD 17	35FD 17	35FD XX	35FD XX
35FE C0	35FE C0	35FE C0	35FE XX
35FF 0A	35FF 0A	35FF 0A	35FF XX
3600 XX	3600 XX	3600 XX	3600 XX
SP 35FB	SP 35FD	SP 35FE	SP 3600

Properties of Well-Written Subroutines

- One Entry Point
- One Exit Point
- One Specific Function
- One Returned Object
- Do NOT store local variables (those used just by your subroutine) at specified memory locations. Use ONLY the stack.



Parameter Passing

- Pass-by-value
 - Much like immediate addressing, the **data itself** is passed.
- Pass-by-reference
 - Much like extended addressing, the **address of the data** is passed.
 - This address must be loaded into an index register to be used by the subroutine.

Questions?

Wrap-up

What we've learned

- Subroutines
- JSR, RTS

What to Come

- Parameter passing