## Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.

Electrical and Computer Engineering

Kettering University

#### Announcement

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.

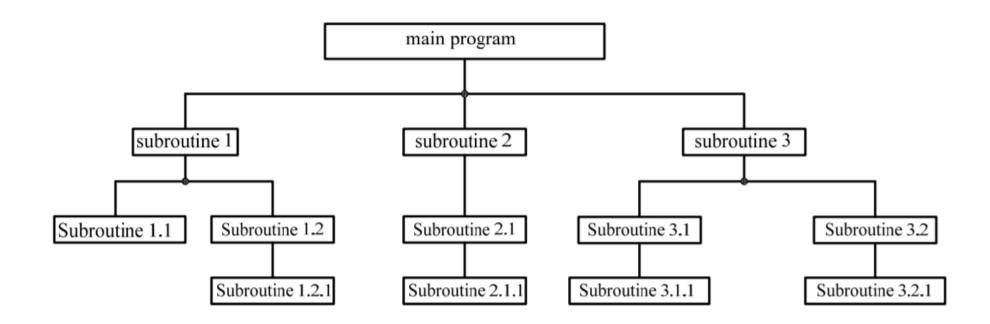
Learn what are pass-by-reference and pass-by-value.

#### 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

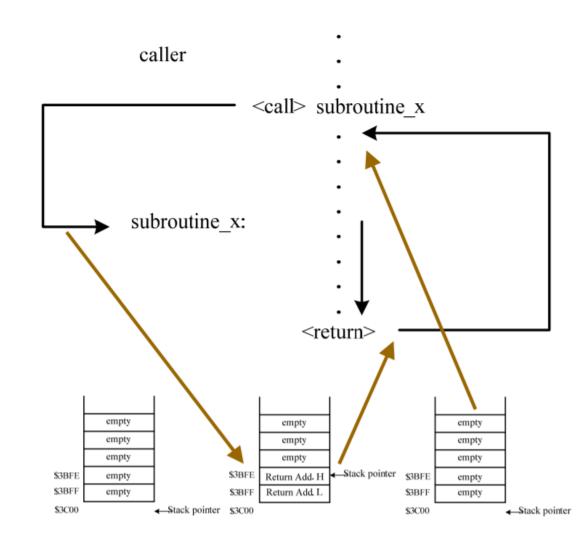
- Subroutine is a sequence of instructions that can be called from many different places in a program
  - One key issue in a subroutine call is to make sure that the program execution returns to the point immediately after the subroutine call when the subroutine completes its computation

- The address for returning to the point immediately after the subroutine completes its computation is known as the <u>return address</u>
  - 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
                           ; C002
                  MagA
                  $1000
                           ; C004
Ret1
         STAA
                           ; C007
         LDAA
                  #-1
         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
                  $C200
         ORG
MagA
         TSTA
         BPL
                  return
         NEGA
         BRA
                  ?????
Return
```

# 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 <u>extended</u> 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

Jsr1 Ret1 Jsr2	ORG LDS LDAA JSR STAA LDAA JSR	\$C000 #\$3600 #17 MagA \$1000 #-1	; C000 ; C003 ; C005 ; C008 ; C00B				
Ret2	STAA	MagA \$1001	; C00D ; C010	After	After	After	After
NOLE	SWI	ψισσι	; C013	Jsr1	Return	Jsr2	Return
; compute magnitude of a single byte number				35FD	35FD	35FD	35FD
•	byte in re			35FE	35FE	35FE	35FE
; output	t: magnitu	ide returne	ed in register A	35FF	35FF	35FF	35FF
MagA		TSTA		3600	3600	3600	3600
		BPL NEGA	return				
Return		RTS		SP	SP	SP	SP

### Example for Subroutines

	ORG	\$C000		
	LDS	#\$3600		; C000
	LDAA	#17		; C003
Jsr1	JSR	MagA		; C005
Ret1	STAA	\$1000		; <b>C008</b>
	LDAA	#-1		; C00B
Jsr2	JSR	MagA		; C00D
Ret2	STAA	\$1001		; <b>C</b> 010
	SWI			; C013
; input: b	yte in reg	gister A	single byte	e number ster A
MagA		TSTA BPL NEGA	return	
Return		RTS		

Afte	er	Afte	er	Afte	er	Afte	er
Jsr1		Ret	urn	Jsr2	<u> </u>	Ret	urn
35FD	XX	35FD	XX	35FD	XX	35FD	XX
35FE	CO	35FE	XX	35FE	CO	35FE	XX
35FF	08	35FF	XX	35FF	10	35FF	XX
3600	XX	3600	XX	3600	XX	3600	XX
SP	35FE	SP	3600	SP	35FE	SP	3600

## **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	1	After MagAB		After eturn		fter HA1
35FB		35F	В	35FB		35FB	
35FC		35F	С	35FC		35FC	
35FD		35FI	D	35FD		35FD	
35FE		35F	E	35FE		35FE	
35FF		35F	F	35FF		35FF	
3600		360	0	3600		3600	
SP		SP		SP		SP	
Aft	er	At	ter	Af	ter	Af	ter
JSI	R1	Re	turn	PU	LA1	RŢ	S1
35FB		35FB		35FB		35FB	
35FC		35FC		35FC		35FC	
35FD		35FD		35FD		35FD	
35FE		35FE		35FE		35FE	
35FF		35FF		35FF		35FF	
3600		3600		3600		3600	
SP		SP		SP [		SP [	

## **Nesting Subroutines**

JsrAB	ORG LDS LDAA LDAB <b>JSR</b> SWI	\$C000 #\$3600 # <b>17</b> #-1 MagAB	; C000 ; C003 ; C005 ; C007 ; C00A
MagAB PSHA1	<b>JSR</b> PSHA	MagA	; C00B ; <b>C00E</b>
1 01 17 (1	TFR	B,A	; C00F
JSR1	JSR	MagA	; C011
	TFR	A,B	; <b>C014</b>
PULA1	PULA		; C016
RTS1	RTS		; C017
MagA	TSTA		; C018
	BPL	return	; C019
	NEGA		; C01B
Return	RTS		; C01C

After JsrAB		Afte <u>Ma</u> g		Afte Ret		Aft PSI	er HA1
35FB	XX	35FB	XX	35FB	XX	35FB	XX
35FC	XX	35FC	CO	35FC	XX	35FC	XX
35FD	XX	35FD	0E	35FD	XX	35FD	17
35FE	CO	35FE	CO	35FE	СО	35FE	CO
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							
	er	Aft	er	Afte	er	Afte	er
	er R1		er urn	Afte PUI		Afte RTS	
JS	R1	Ret	urn	PUI	_A1	RTS	1
JS 35FB	R1	Ret 35FB	XX	PUI 35FB	_A1	RTS 35FB	XX
JS 35FB 35FC	C0 14	Ret 35FB 35FC	XX XX	PUI 35FB 35FC	XX XX	RTS 35FB 35FC	XX XX
JS 35FB 35FC 35FD	C0 14 17	Ret 35FB 35FC 35FD	XX XX 17	9UI 35FB 35FC 35FD	XX XX XX	35FB 35FC 35FD	XX XX XX
35FB 35FC 35FD 35FE	C0 14 17 C0	Ret 35FB 35FC 35FD 35FE	XX XX 17 C0	35FB 35FC 35FD 35FE	XX XX XX CO	35FB 35FC 35FD 35FE	XX XX XX XX
35FB 35FC 35FD 35FE 35FF	C0 14 17 C0 0A	Ret 35FB 35FC 35FD 35FE 35FF	XX XX 17 CO OA	35FB 35FC 35FD 35FE 35FF	XX XX XX CO OA	35FB 35FC 35FD 35FE 35FF	XX XX XX XX XX

#### Properties of Well-Written Subroutines

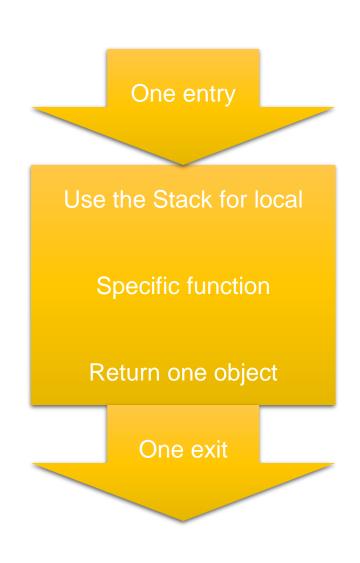
One Entry Point

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 <u>immediate addressing</u>, 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