

Mark II Aiken Relay Calculator

Recap

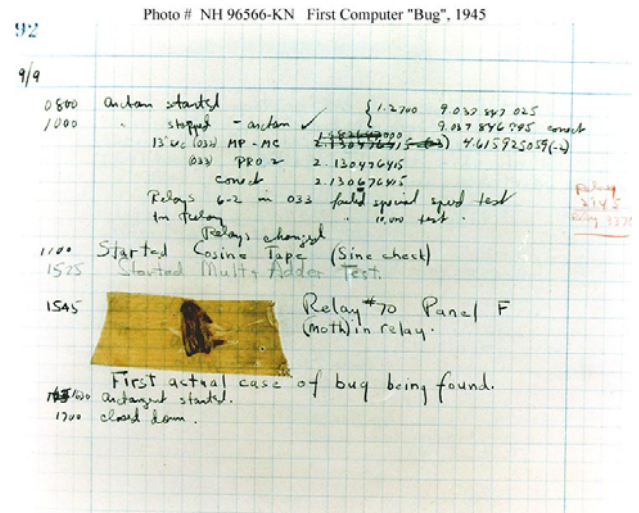
- 9S12 Architecture, registers
- Execution thinking about simplified bus cycles
- Memory map: I/O, RAM, EEPROM

Overview

- Continuation of execution
- Stack
- Subroutines
- Parallel port, direction registers

Start with first question of Worksheet 5

Question 1. What are the six phases of execution?



2.4. Simplified 9S12 Machine Language Execution

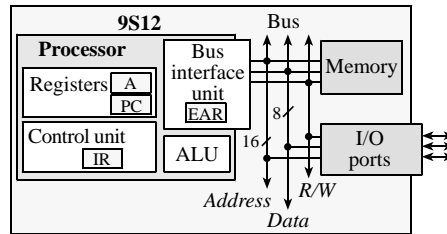


Figure 2.6. Block diagram of a simplified 9S12 computer.

The bus interface unit (BIU)

- reads data from the bus during a read cycle,
- writes data onto the bus during a write cycle.
- always drives the address bus and the control signals
- effective address register (EAR)** contains the data address

The control unit (CU) (EE306, EE360M, EE360N)

- orchestrates the sequence of operations
- issues commands to ALU, BIU
- instruction register (IR)** contains the op code

The registers

- high-speed storage devices located in the processor
- do not have addresses like regular memory
- specific functions explicitly defined by the instruction
- Accumulators** contain data (A, B, D)
- Index registers** contain addresses (X, Y)
- Program counter (PC)** points to instruction to execute next
- Stack pointer (SP)** points to the top element on the stack
 - context switch when calling and returning from a function

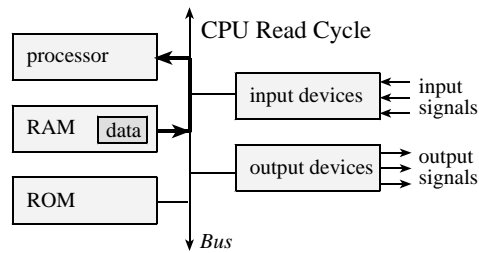
- pass parameters
- save temporary information
- implement local variables
- **Condition code register (CCR)** the status of the previous operation

The arithmetic logic unit (ALU)

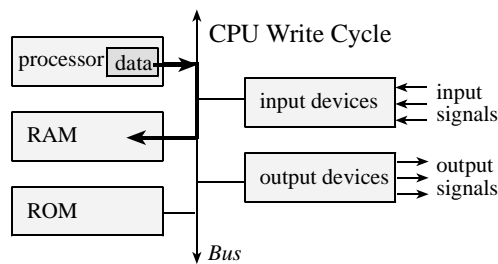
- Arithmetic operations
 - Addition
 - Subtraction
 - Multiplication
 - Division
- Logic operations
 - And
 - Or
 - Exclusive or
 - Shift

The bus

- address *where or which module*
- data *what*
- control *when and direction*



A read cycle copies data from RAM, ROM or input device into the processor.



A write cycle copies data from the processor into RAM, or output device.

<u>Phase</u>	<u>Function</u>	<u>R/W</u>	<u>Address</u>	<u>Comment</u>
1	Op code fetch	read	PC++	Put op code into IR
	Operand fetch	read	PC++	Immediate or calculate EA
2	Decode instruction	none		Figure out what to do
3	Evaluation address	none		Determine EAR
4	Data read	read	SP,EAR	Data passes through ALU,
5	Free cycle	read	PC/SP/\$FFFF	ALU operations, set CCR
6	Data store	write	SP,EAR	Results stored in memory

Question 2. Assume this listing file output
\$5000 B60258 ldaa \$0258

- Part a) What addressing mode is it?
- Part b) What happens when it executes?
- Part c) Show the simplified bus cycles as it executes

LDAA Load Accumulator A LDAA

Operation: (M) ⇒ A
Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

Condition Codes and Boolean Formulas:

S	X	H	I	N	Z	V	C
-	-	-	-	Δ	Δ	0	-

N: Set if MSB of result is set; cleared otherwise.
 Z: Set if result is \$00; cleared otherwise.
 V: 0; Cleared.

Addressing Modes, Machine Code, and Execution Times:

Source Form	Address Mode	Object Code	Cycles	Access Detail
LDAA #opr <i>b</i>	IMM	86 i1	1	P
LDAA opr <i>b</i> a	DIR	96 dd	3	rFP
LDAA opr16 <i>a</i>	EXT	B6 hh i1	3	rOP
LDAA opr <i>x0</i> ,xy <i>sp</i>	IDX	A6 xb	3	rFP
LDAA opr <i>x9</i> ,xy <i>sp</i>	IDX1	A6 xb ff	3	rPO
LDAA opr <i>x16</i> ,xy <i>sp</i>	IDX2	A6 xb ee ff	4	rPP
LDAA [D,xy <i>sp</i>]	[D,IDX]	A6 xb	6	rFP
LDAA [opr <i>x16</i> ,xy <i>sp</i>]	[IDX2]	A6 xb ee ff	6	rFP

Notgate example from Lecture 2

Action: Start TExaS, open NotGate2.uc
 Assemble, tile windows
Observe: See listing file, explain components
 Address Data Instructions

```

;not gate 8/28/2008 9:01:38 PM
$0258                    PTP        equ    $0258        ;Port P I/O
$025A                    DDRP       equ    $025A        ;Direction
$0242                    DDRT       equ    $0242        ;Direction
$0240                    PTT        equ    $0240        ;Port T I/O
$0800                    org    $0800

$4000                    org    $4000
$4000    CF4000            main    lds    #$4000
$4003    86FF                    ldaa   #$FF
$4005    7A025A                  staa   DDRP
$4008    8600                    ldaa   #$00
$400A    7A0242                  staa   DDRT
$400D    10EF                    cli                    ;debugger
$400F    B60240            loop    ldaa   PTT            ;input
$4012    8880                    eora   #$80            ;not gate
$4014    7A0258                  staa   PTP            ;output
$4017    20F6                    bra    loop

$FFFE                    org    $FFFE
$FFFE    4000             fdb    main
    
```

Draw a matrix showing PC, A, SP, IR, EAR, PTP, PTT
 Draw a memory model of this system
 Hand execute, showing simplified bus cycles
 Action: Single step and compare to table

Do another example of a relative branch

Assume bra there is at \$5000
 Assume there is at \$5036
 What is machine code?

Why doesn't this relative branch work

Assume bra there is at \$6000
 Assume there is at \$6096
 What is machine code?

How to fix this?

lbra there ; uses 16-bit relative addressing
 jmp there ; uses 16-bit extended mode addressing

Stack

```

$4000
$4000 CF4000          main    org  $4000
$4003 8601            lds  #$4000
$4005 36              ldaa #1
$4006 8602            psha
$4008 36              ldaa #2
$4009 8603            psha
$400B 36              ldaa #3
                        psha
    
```

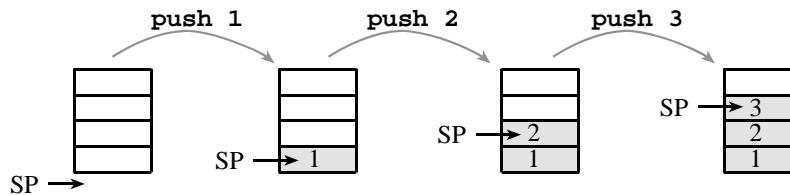


Figure 2.5. Stack picture as three numbers are pushed.

Draw a matrix showing PC, A, SP, IR, EAR
 Draw a memory model of this system
 Hand execute up to first psha, showing simplified bus cycles

2.8. Subroutines

functions, which return values,
procedures, which do not
subroutine for all subprograms whether or not they return a value.

```

$0800                                org  $0800
    
```

```

$0800          Flag rmb 1
$0801          Data rmb 2
$4000
                org $4000
                ;*****Set*****
                ; Set Data=1000, and Flag=1
                ; Input: None
                ; Output: None
$4000 180303E80801 Set movw #1000,Data ;3
$4006 180B010800   movb #1,Flag ;4
$400B 3D          rts ;5
$400C CF4000      main lds #$4000 ;1
$400F 07EF       bsr Set ;2
$4011 20FE       loop bra loop ;6
$FFFE           org $fffe
$FFFE 400C       fdb main
    
```

Program 2.1. Listing file showing how to use the **bsr** and **rts** instructions to implement a subroutine.

Draw a matrix showing PC, SP, IR, EAR

Draw a memory model of this system

Hand execute up to first psha, showing simplified bus cycles

Opcode fetch R 0x400F 0x07 from ROM Phase 1
 Operand fetch R 0x4010 0xEF from ROM Phase 1
 Stack store lsbW 0x3FFF 0x11 to RAM Phase 6
 Stack store msbW 0x3FFE 0x40 to RAM Phase 6

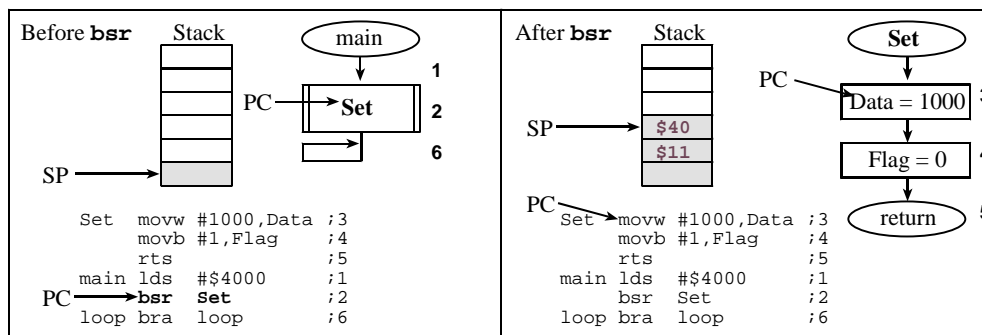


Figure 2.12. The stack before and after execution of the **bsr** instruction.

Opcode fetch R 0x4009 0x3D from ROM Phase 1
 Stack read msb R 0x3FFE 0x40 from RAM Phase 4
 Stack read lsb R 0x3FFF 0x06 from RAM Phase 4

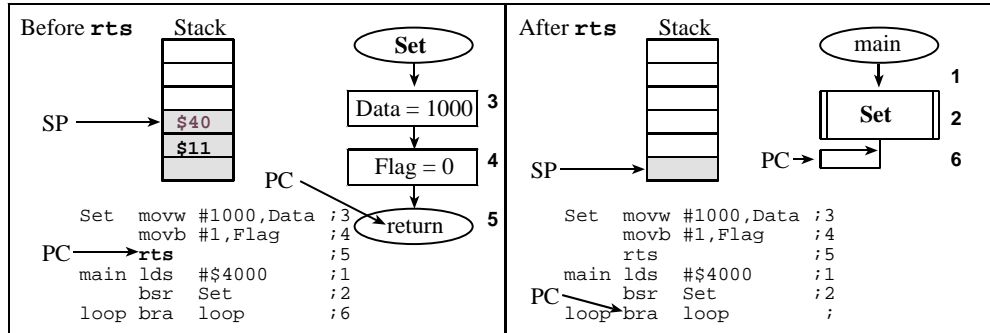


Figure 2.13. The stack before and after execution of the *rts* instruction.

2.9. Input/Output

I/O ports

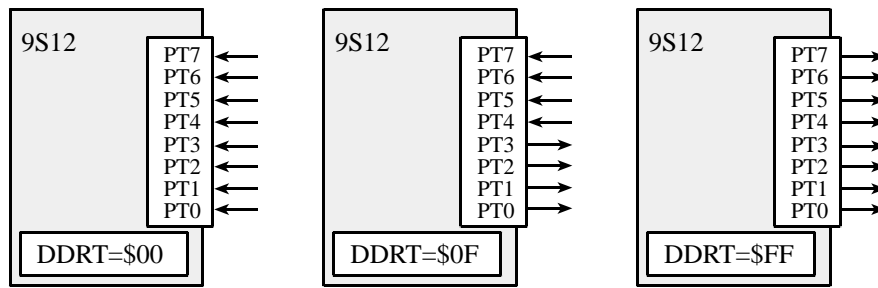


Figure 2.14. The input/output direction of a bidirectional port is specified by its direction register.

DDRH, DDRP, DDRJ, DDRT, specify if corresponding pin

0 means input

1 means output

Where to find addresses for PTH DDRH?

Book, Chapter 4, Program 4.3 (should have been in the index)

9S12DP512 data sheet

Port12.rtf file as part of TExaS install

My favorite is the TExaS help system

Question 3. Write code to make Port H bits 7,5,3,1 output, bits 6,4,2,0 input

The bottom line

Computer executes one instruction at a time

Stack is used for temporary data, return address

Subroutines allow for modular programming

I/O ports allow data to flow into/out of computer

Usually, we set DDR once at the beginning