

6811 Programming Model

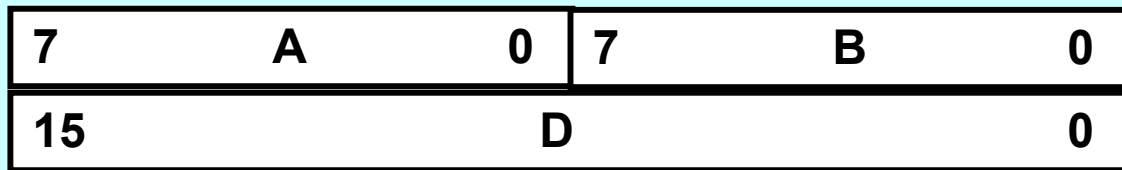
Branch Instructions

Condition Code Register

Comparison Instructions

Review 1

Simplified



8-bit Accumulators A & B

16-bit Accumulator D



Program Counter



Condition Code Register

Carry/borrow (MSB)

oVerflow (2s C)

Zero

Negative

Half carry (bit 3→4),(BCD)

Review 2

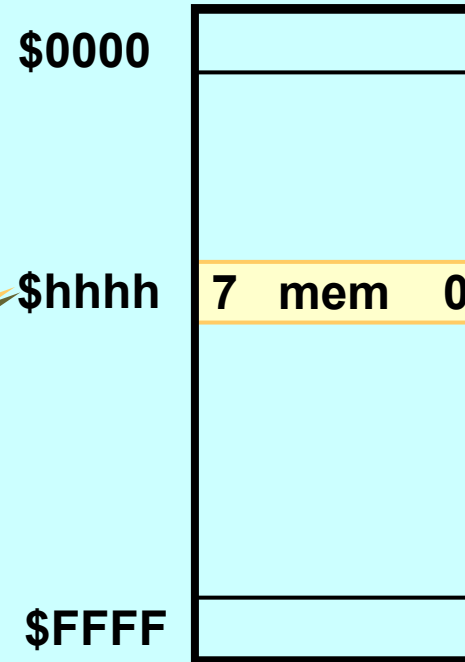
Memory Model

Memory = 64K x 8

Number Notation

\$ denotes hexadecimal

$h = 0000 \dots 1111$



15 PC 0

16-bit Program Counter

7 ACCx 0

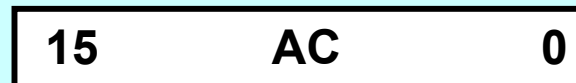
8-bit Accumulator

Review 3

Simple Fictional Machine - Registers

Mnemonic	Operation	Op-code
LOAD	mem<qhhh> -> AC	00
STORE	AC -> mem<qhhh>	01
ADD	AC + mem<qhhh> -> AC	10
BRN	IF AC<15> =1 THEN qhhh -> PC	11

What registers do we need?



16-bit Accumulator



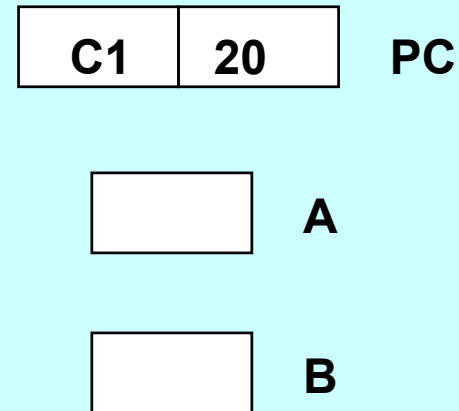
14-bit Program Counter

Review 4: N1 + N2 → SUM

6811 Program fragment

LDA	\$C120	B6
	\$C121	C2
	\$C122	10
ADD	\$C123	BB
	\$C124	C2
	\$C125	11
STA	\$C126	B7
	\$C127	C2
	\$C128	12
HALT	\$C129	3F

N1	\$C210	12
N2	\$C211	34
SUM	\$C212	FF



Let's go through it
instruction by instruction

Branch Instructions

Examples:

BEQ Branch on Equal to Zero (i.e., Z bit = 1)
if(Z == 1) go to <a_specified_address>;

BRA Branch Always
go to <a_specified_address>;

JMP Jump Always
go to <a_specified_address>;

This is
an example of a
conditional branch
instruction

These are examples of
unconditional branch
instructions

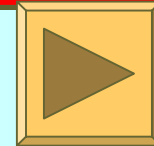
Now, how are these
two different?
Look in the PRG!

What's Different?

Source Form	Operation	Boolean Expression	Addressing Mode for Operand	Machine Code		Bytes
				Opcode	Operand	
BRA	Branch Always	?1=1	REL	20	rr	2
JMP	Jump	See Special Operations	EXT	7E	hh ll	3
			IND,X	6E	ff	2
			IND,Y	18 6E	ff	3

This is a new addressing mode
Relative addressing

Relative to what?



Relative Addressing

In this mode, addresses are specified **relative to the program counter**

Also known as **PC-relative addressing**

Example:

\$6142 BRA \$28



**Relative
Offset**

Program branches to address (PC + \$28)

To figure out this address, we need to know the value in the PC.

Where is the instruction located in memory?

BRA is a 2-byte instruction.

After the instruction is fetched, **PC = \$6142 + \$02 = \$6144.**

Therefore, **PC + \$28 = \$6142 + \$28 = \$616A.**

Backward Jumps

The relative offset is treated as a **signed number** in two's complement notation

Example:

\$6142 BRA \$FE

PC = \$6144

Relative offset = \$FE

Extend the offset to 16 bits → \$FFFE

PC + relative offset = \$6144 + \$FFFE = \$6142

Its an infinite loop!!

What does this say about this instruction?

	0110	0001	0100	0100
+	1111	1111	1111	1110
<hr/>				
	0110	0001	0100	0010

Inherent Addressing Mode

Instructions with No Operands

Some of the instructions that have no operand:

ABA $A + B \rightarrow A$

CLRB $0 \rightarrow B$

INCA $A + 1 \rightarrow A$

DECB $B - 1 \rightarrow B$

SEC $1 \rightarrow C$ ($C = \text{CCR}\langle 0 \rangle$)

Lec. Exercise 1

Translate the following pseudo-code program into M6811 machine code. The program starts at address \$6000. Use the SWI instruction to HALT the program.

\$B000 → IX;

\$0A → ACCA;

L1: ACCA - 1 → ACCA;

IX - 1 → IX;

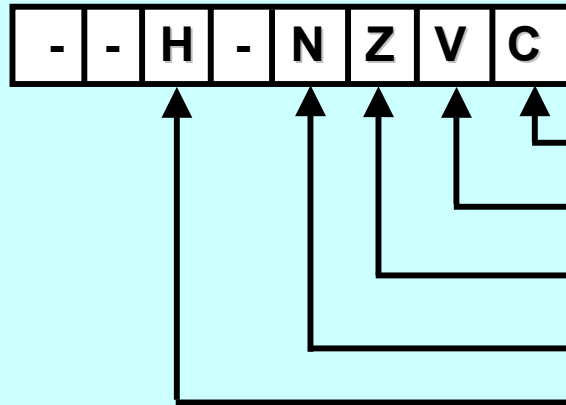
\$35 → M(IX + 0);

if (ACCA ≠ 0) then go to L1 else HALT;

Exercise 1 Answer

<u>Address</u>	<u>Contents</u>	<u>Comments</u>
\$6000	CE	LDX (imm): \$B000 → IX
\$6001	B0	
\$6002	00	
\$6003	86	LDAA (imm): \$0A → ACCA
\$6004	0A	
\$6005	4A	DECA (inh): L1: ACCA - 1 → ACCA
\$6006	09	DEX (inh): IX - 1 → IX
\$6007	C6	LDAB (imm): \$35 → ACCB
\$6008	35	
\$6009	E7	STAB (ind,X): ACCB → M(IX + 0)
\$600A	00	
\$600B	4D	TSTA (inh): ACCA - 0
\$600C	26	BNE (rel): if (ACCA ≠ 0) then go to L1
\$600D	F7	else HALT
\$600E	3F	"HALT"

Condition Code Register



Condition Code Register

Carry/borrow (MSB)

oVerflow (2s C)

Zero

Negative

Half carry from (bit 3 to 4), used for (BCD) corrections only

These CCR bits are set by
ALU operations

Simple Branches

Test only 1 or 0

Source Form	Operation	Boolean Expression	Opcode
BRA (opr)	Branch Always	?1 = 1	20
BRN (opr)	Branch Never	?1 = 0	21

Simple Conditional Branches

Test only one of N, Z, V, C

Source Form	Operation	Boolean Expression	Opcode
BMI (opr)	Branch if Minus	?N = 1	20
BPL (opr)	Branch if Plus	?N = 0	2A
BEQ (opr)	Branch if Equal (zero)	?Z = 1	27
BNE (opr)	Branch if Not Equal	?Z = 0	26
BVS (opr)	Branch if oVerflow Set	?V = 1	29
BVC (opr)	Branch if oVerflow Clear	?V = 0	28
BCS (opr)	Branch if Carry Set	?C = 1	25
BCC (opr)	Branch if Carry Clear	?C = 0	24

Signed Conditional Branches

Test combinations of N, Z and V

Source Form	Operation	Boolean Expression	Opcode
BGT (opr)	Branch if > Zero	$?Z + (N \oplus V) = 0$	2E
BLE (opr)	Branch if <= Zero	$?Z + (N \oplus V) = 1$	2F
BGE (opr)	Branch if >= Zero	$?N \oplus V = 0$	2C
BLT (opr)	Branch if < Zero	$?N \oplus V = 1$	2D

Unsigned Conditional Branches

Test combinations of Z and C

Source Form	Operation	Boolean Expression	Opcode
BHI (opr)	Branch if Higher	?C + Z = 0	22
BLS (opr)	Branch if Lower or Same	?C + Z = 1	23
BHS (opr)	Branch if Higher or Same	?C = 0	24*
BLO (opr)	Branch if Lower	?C = 1	25**

* Same Opcode as BCC

** Same Opcode as BCS

Other Branch Instructions

Bit Manipulation Branches

Source Form	Operation	Boolean Expression	Addr Mode	Opcode
BRCLR (opr)(msk)(rel)	Branch if bit(s) Clear	$?M \cdot mm = 0$	DIR IND,X IND,Y	13 1F 18 1F
BRSET (opr)(msk)(rel)	Branch if bit(s) Set	$? \bar{M} \cdot mm = 0$	DIR IND,X IND,Y	14 1C 18 1C

Branch to Subroutine

BSR (opr)	Branch to Subroutine	Special*	REL	8D
-----------	----------------------	----------	-----	----

* See Special Operations page in the PRG

Comparison Instructions

8 Bit Comparisons

Source Form	Operation	Boolean Expression	Addr Mode	Opcode
CMPA (opr)	Compare A to Memory	A - M	IMM	81
			DIR	91
			EXT	B1
			IND,X	A1
			IND,Y	18 A1
CMPB (opr)	Compare B to Memory	B - M	IMM	C1
			DIR	D1
			EXT	F1
			IND,X	E1
			IND,Y	18 E1

Comparison Instructions

16 Bit Comparisons

Source Form	Operation	Boolean Expression	Addr Mode	Opcode
CPD (opr)	Compare D to Memory	D - M:M+1	IMM	1A 83
			DIR	1A 93
			EXT	1A B3
			IND,X	1A A3
			IND,Y	CD A3
CPX (opr)	Compare X to Memory	X - M:M+1	IMM	8C
			DIR	9C
			EXT	BC
			IND,X	AC
			IND,Y	CD AC
CPY (opr)	Compare Y to Memory	Y - M:M+1		

Lec. Exercise 2

For this program fragment, determine the Branch instruction that checks " $N1 \leq N2$ " for both cases.

```
START  LDAA N1
        CMPA N2
        B??  DONE

MORE   ...

DONE   ...
```

Suppose $N1 = \$42$ and $N2 = \$BA$.

Case 1. $N1$ and $N2$ are unsigned numbers

Case 2. $N1$ and $N2$ are signed numbers

Exercise 2 Answer

```
START  LDAA N1
        CMPA N2
        B??  DONE
MORE   ...
```

Suppose
N1 = \$42 = 68 and
N2 = \$BA 186 or -70.

The Branch instruction that checks $?N1 \leq N2$:

Case 1. N1 and N2 are unsigned numbers

BLS

Case 2. N1 and N2 are signed numbers

BLE

Stacks

Subroutines

Complete Programmer's Model

7	A	0	7	B	0
15	D				0
15	IX				0
15	IY				0
15	PC				0
15	SP				0

8-bit Accumulators A & B

16-bit Accumulator D

16-bit Index Register IX

16-bit Index Register IY

Program Counter

Stack Pointer



Condition Code Register

Carry/borrow (MSB)

Overflow (2s C)

Zero

Negative

Half carry (bit 3→4),(BCD)

STOP disable

X-interrupt mask

I-interrupt mask

Stacks

Stacks

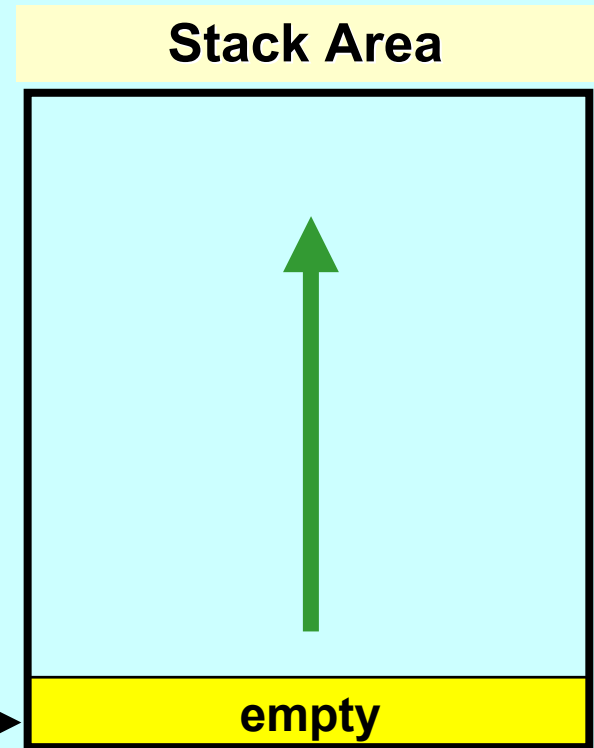
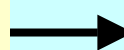
Grow down, **toward low memory**

Data

PuSHed on the stack (**PSH**)

PULled off the stack (**PUL**)

Stack Pointer

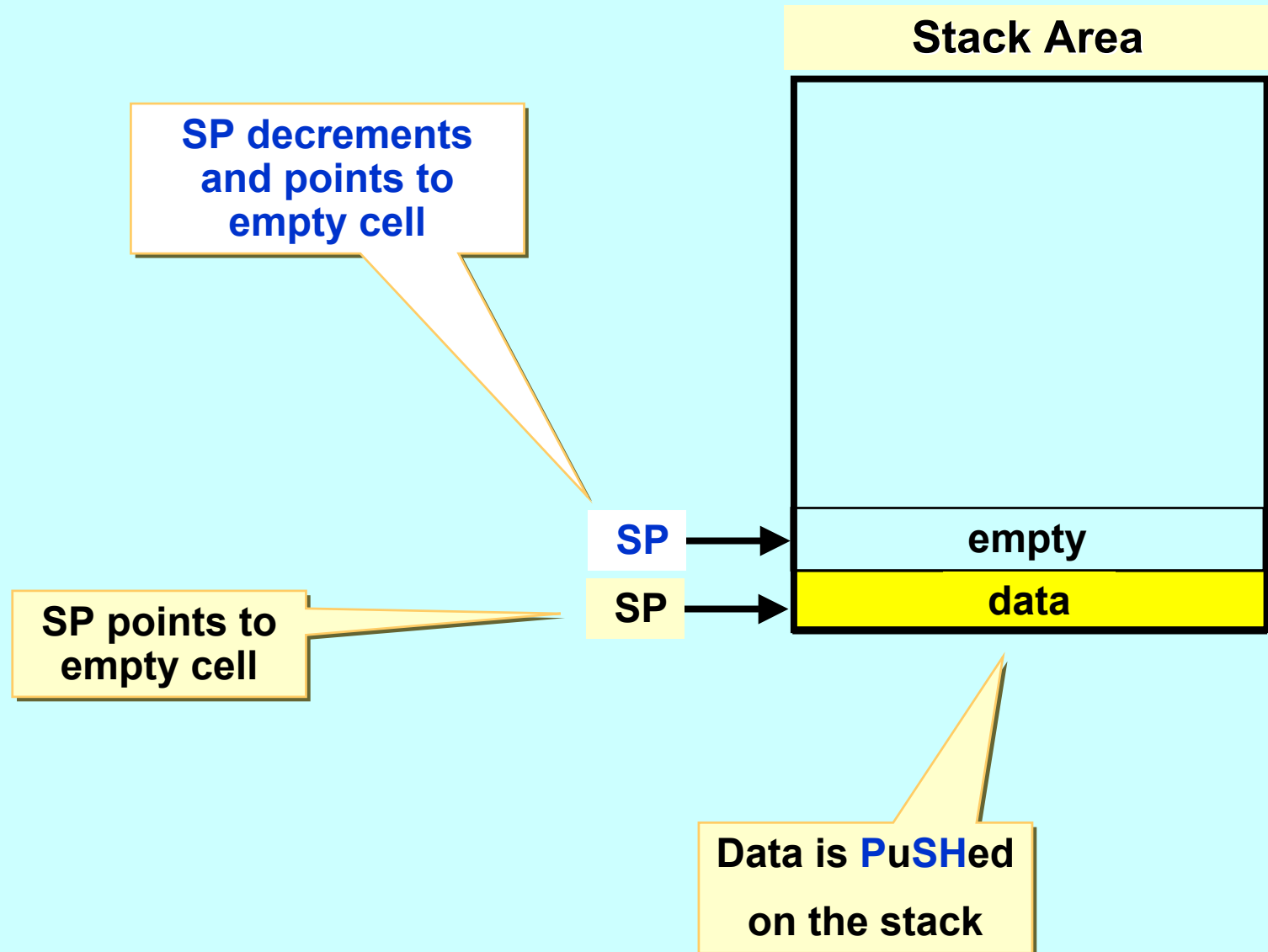


Always **PUL** as many times as **PSH**

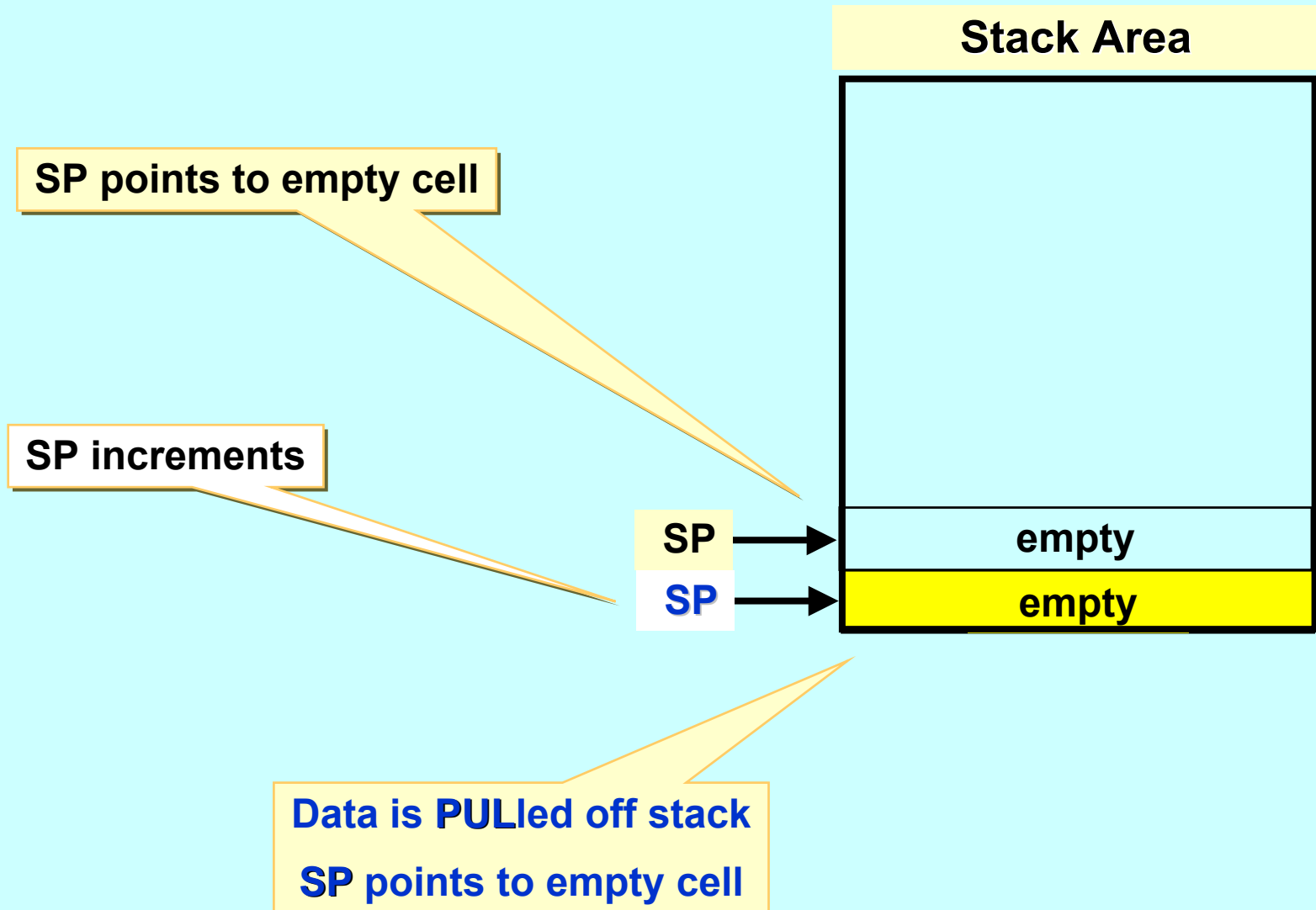
15	SP	0
----	----	---

The Stack Pointer always points to the next empty cell on the stack

PuSHing Data on Stacks



PULLing Data off Stacks



Stack Instructions

Reserve a Stack Area, e.g., 256 cells:

```
STACK RMB 256
```

```
INITSP EQU *-1
```

Top of Stack

Used
together

Initialize the Stack Pointer:

```
INIT LDS #INITSP
```

Push Registers on Stack:

```
PSHA Push ACCA
```

```
PSHB Push ACCB
```

```
PSHX Push IX (2 bytes)
```

```
PSHY Push IY (2 bytes)
```

Similarly, PULA, PULB, PULX, and PULY

More Stack Instructions

Transfer Registers:

TSX $SP+1 \rightarrow X$

TSY $SP+1 \rightarrow Y$

TXS $IX-1 \rightarrow SP$

TYS $IY-1 \rightarrow SP$

Notice the
increment SP and decrement IX, IY

Increment / Decrement Stack Pointer:

INS $SP+1 \rightarrow SP$

DES $SP-1 \rightarrow SP$

Lots of addressing
modes !!

Load / Store Stack Pointer:

LDS $M:M+1 \rightarrow SP$ IMM, DIR, EXT, indX, indY

STS $SP \rightarrow M:M+1$ DIR, EXT, indX, indY

Lec. Exercise 1

A. Write an assembly language program starting at \$6A00 that stores the contents of the A,B,X,Y, and CCR registers on the stack.

(A) = \$2B, (B) = %00001001, (X) = \$6F00, (Y) = \$C000, and (CCR) = %01x0xxxx.

B. Write an assembly language program starting at \$D300 that loads the contents of the A,B,X,Y, and CCR registers from the stack.

(SP) = \$01, (SP+1) = \$02, (SP+2) = \$03, (SP+3) = \$04, (SP+4) = \$05, (SP+5) = \$06, (SP+6) = \$07, (SP+7) = \$08.

Exercise 1 Answer

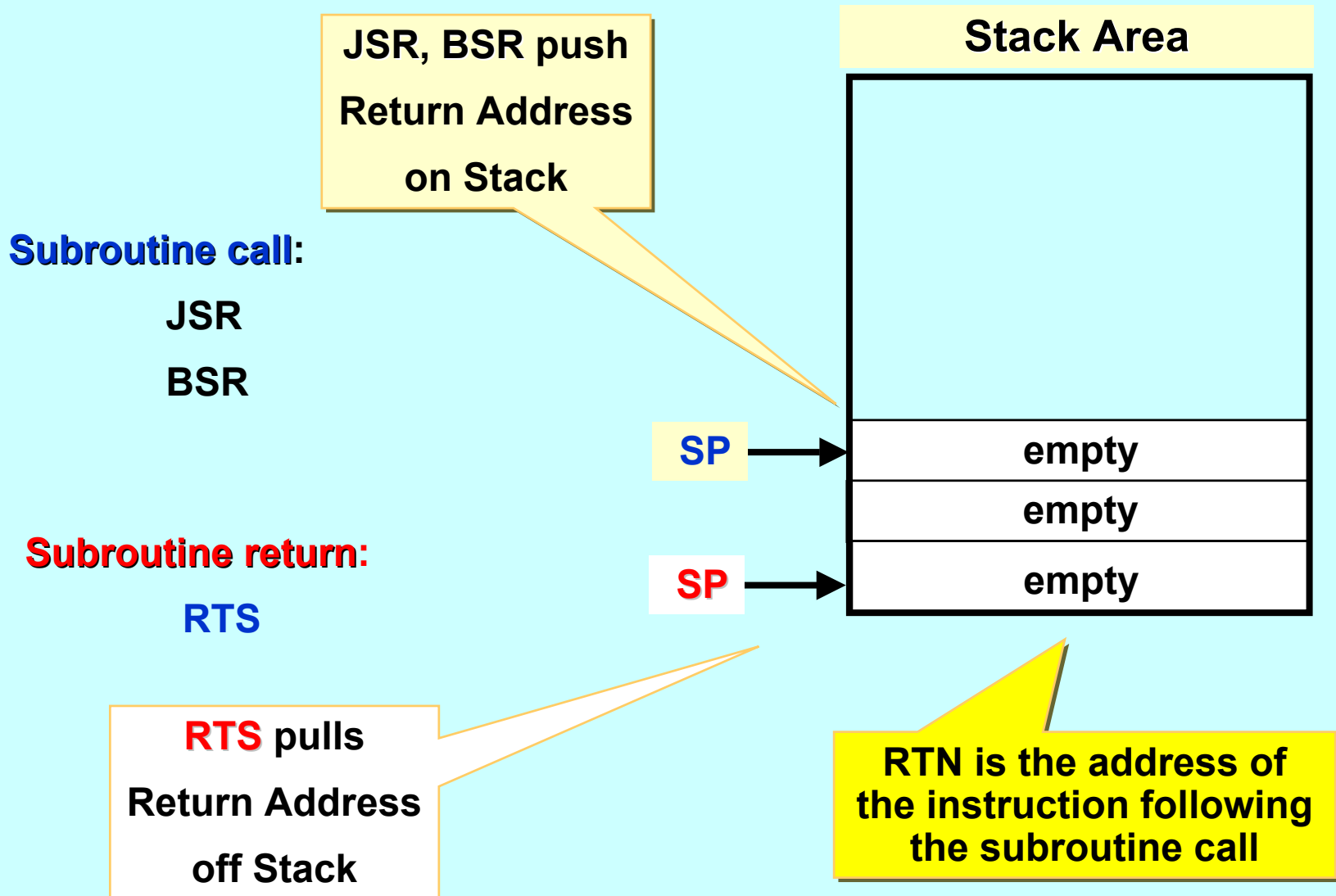
A. Push A,B,X,Y,CCR on Stack (in that order)

\$6A00	BE	LDS #\$004A	Buffalo User Stack area
\$6A01	00		
\$6A02	4A		
\$6A03	36	PSHA	
\$6A04	37	PSHB	
\$6A05	3C	PSHX	
\$6A06	18	PSHY	
\$6A07	3C		
\$6A08	07	TPA	use ACCA
\$6A09	36	PSHA	
\$6A0A	3F	HALT	

B. Pull A,B,X,Y,CCR off Stack

\$D300	32	PULA	
\$D301	33	PULB	
\$D302	38	PULX	
\$D303	18	PULY	
\$D304	B7	STAA SAVEA	
\$D305	D3		
\$D306	0D		
\$D307	32	PULA	get CCR
\$D308	06	TAP	
\$D309	B6	LDAA SAVEA	
\$D30A	D3		
\$D30B	0D		
\$D30C	3F	HALT	
\$D30D	--	(SAVEA)	holds ACCA

Subroutine Instructions



Subroutine Arguments

Pass by value:

Copy data to a register or location known to the subroutine

Accumulators are commonly used to pass values

Pass by reference:

Provide a pointer to the data location

Index registers are commonly used to pass pointers

Global variables:

All variables, and constants, are global when using the Motorola cross-assembler

All source code is in the same file; hence labeled variables and constants are all in the same symbol table

Lec. Exercise 2

Write a subroutine to reverse the order of a list of length LISTLEN and pointed to by the X-register.

Make a flowchart or pseudocode version first!

- * SUBROUTINE REVERSE
- * REVERSES A LIST OF LENGTH 'LISTLEN'
- * X POINTS TO THE LIST

...

...

RTS

- * SUBROUTINE DATA SECTION

TEMP RMB 1

Exercise 2 Answer P-C

Pseudo Code for

- * SUBROUTINE REVERSE
- * REVERSES A LIST OF LENGTH 'LISTLEN'
- * X POINTS TO THE LIST

Subroutine setup:

**X points to front of list, LISTLEN contains
number of elements in the list**

LISTLEN/2 → COUNT; truncates if LISTLEN is odd

X + LISTLEN - 1 → Y; Y points to end of list

for 1 to COUNT do

M(X) → A; swap M(X) and M(Y)

M(Y) → B;

A → M(Y);

B → M(X);

X + 1 → X; move pointers toward middle

Y - 1 → Y;

return;

Exercise 2 Answer Asm

- * SUBROUTINE REVERSE
- * REVERSES A LIST OF LENGTH 'LISTLEN'
- * X POINTS TO THE LIST

```

                ORG    $7000
REVERSE        EQU    *
                LDAB  LISTLEN
                ASRB
                STAB  COUNT
                STX   XSTORE
                LDY   XSTORE
                LDAB  LISTLEN
                ABY
                DEY

                LOOP  LDAA  0,X
                    LDAB  0,Y
                    STAB  0,X
                    STAA  0,Y
                    INX
                    DEY
                    DEC   COUNT
                    BNE  LOOP
                    RTS

                COUNT RMB  1
                XSTORE RMB  2
                END
```