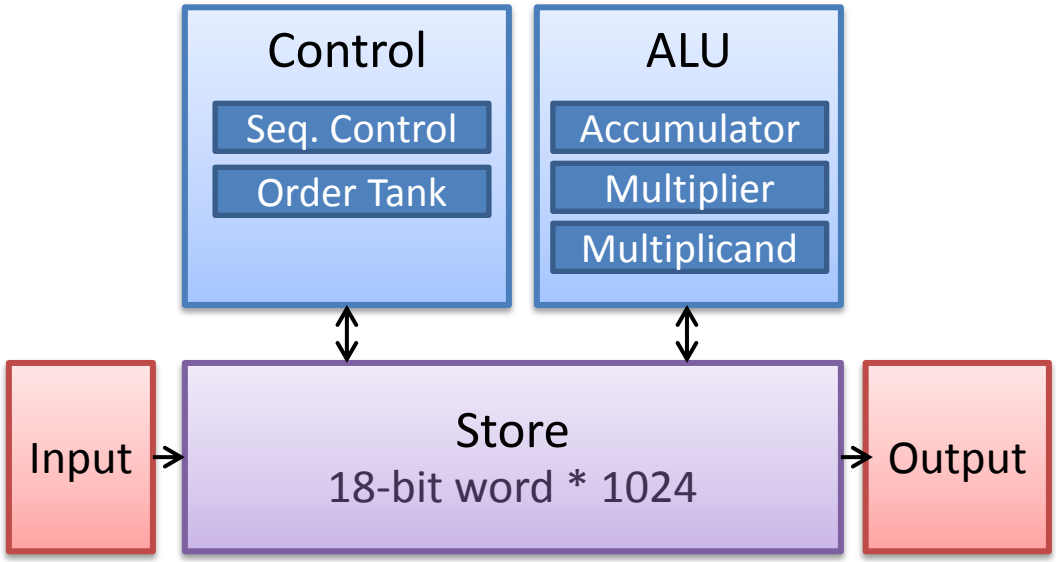
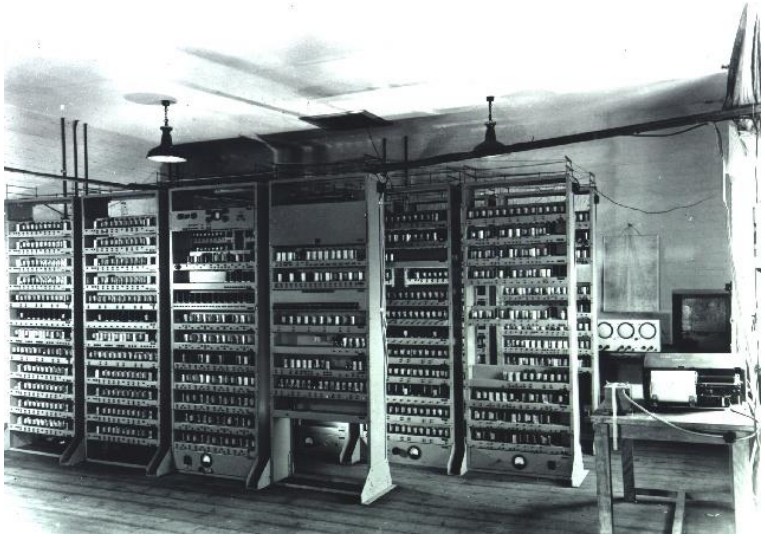
EDSACjr and Self Modifying Code

TA: Hsin-Jung Yang

Cambridge EDSAC (1949)

- **EDSAC:** Electronic Delay Storage Automatic Calculator





- Single accumulator + absolute addressing of memory
- Typical execution time: ~600 simple commands/sec

Ref1: http://en.wikipedia.org/wiki/Electronic_Delay_Storage_Automatic_Calculator
Ref2: http://www.dcs.warwick.ac.uk/~edsac/Software/EdsacTG.pdf

EDSACjr

- A simplified version of EDSAC instruction set

Opcode	Description	Bit Representation
ADD n	Accum ← Accum + M[n]	00001 n
SUB n	Accum ← Accum - M[n]	10000 n
STORE n	M[n] ← Accum	00010 n
CLEAR	Accum ← 0	00011 00000000000
OR n	Accum ← Accum | M[n]	00000 n
AND n	Accum ← Accum & M[n]	00100 n
SHIFTR n	Accum ← Accum shiftr n	00101 n
SHIFTL n	Accum ← Accum shiftl n	00110 n
BGE n	If Accum ≥ 0 then PC ← n	00111 n
BLT n	If Accum < 0 then PC ← n	01000 n
END	Halt machine	01010 00000000000

M[x]: the contents of the memory location addressed by x

Instruction Format:

Opcode	Immediate
5 bit	11 bit

Programming with EDSACjr

- **Macro:** a sequence of instructions defined with a short name and is expanded in place at assembly time
- A macro is not the same as a subroutine or function
- Example:

Macro Definition

```
.macro LOAD (n)
        CLEAR
        ADD n
.end macro
```

Program

```
LOAD A
SUB B
STORE B
LOAD A
```

expand →

```
CLEAR
ADD A
SUB B
STORE B
CLEAR
ADD A
```

Programming with EDSACjr

- **Global label:** a label defined outside the macro
- Global labels are useful for accessing commonly used memory locations
- Example:

Macro Definition

```
.macro STOPGE
        BGE  stop
.end macro
```

Program

```
        STOPGE
        SUB B
stop:   ADD A
```

expand →

```
        BGE stop
        SUB B
stop:   ADD A
```

Programming with EDSACjr

- **Local label:** a label defined within the macro
- Local labels will be replaced by unique label names during expansion
- Example1:

Macro Definition

```
.macro HANGGE
        here: BGE here
.end macro
```

Program

```
HANGE
SUB B
HANGE
```

expand →

```
here_1: BGE here_1
SUB B
here_2: BGE here_2
```

Programming with EDSACjr

- **Local label:** a label defined within the macro
- Local labels will be replaced by unique label names during expansion
- Example2:

Macro Definition

```
.macro ADDGE n
        BLT cont
        ADD n
    cont:
.end macro
```

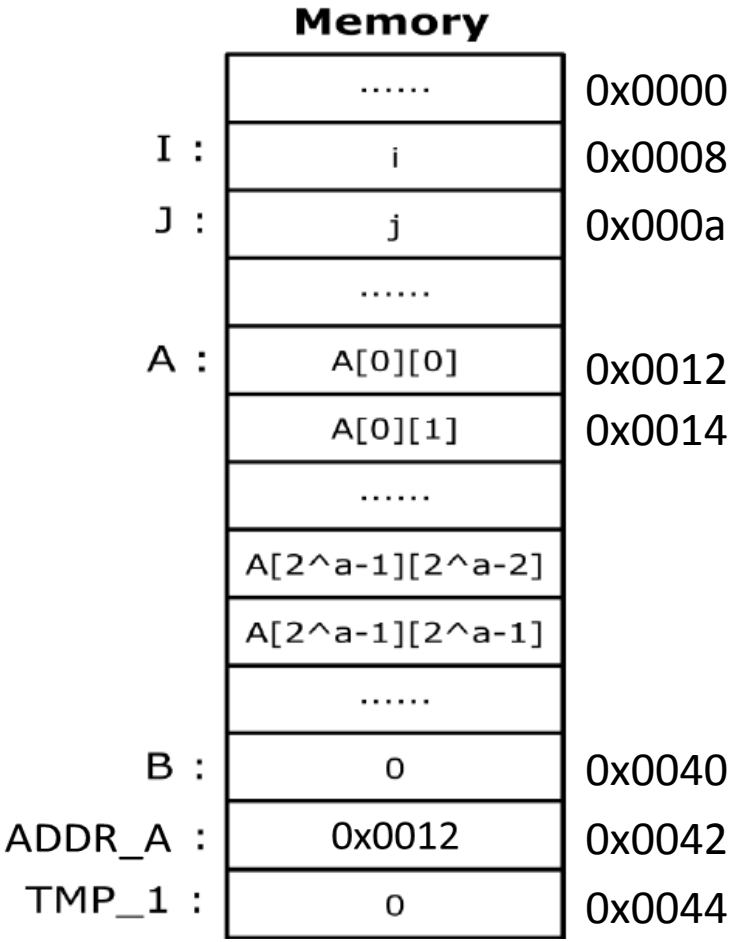
Program

```
ADDGE A
SUB B
```

expand →

```
        BLT cont_1
        ADD A
cont_1: SUB B
```

Self Modifying Code Example



Global variables

```
_store_op:  STORE 0     ; STORE template
_bge_op:    BGE 0       ; BGE template
_blt_op:    BLT 0       ; BLT template
_add_op:    ADD 0       ; ADD template
```

Pseudo-code:

B = A[i][j] + A[j][i]

A is a 2^a by 2^a matrix where a is a constant.

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
CLEAR
ADD      I
```

Calculate address offset: 2^a*i + j

i

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
CLEAR
ADD      I
SHIFTL   a
```

Calculate address offset: 2^a*i + j

2^a*i

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
CLEAR
ADD     I
SHIFTL  a
ADD     J
```

Calculate address offset: 2^a*i + j

2^a*i+j

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
CLEAR
ADD      I
SHIFTL   a
ADD      J
ADD      ADDR_A
```

address of A[i][j]

0x0012+2^a*i+j

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
CLEAR
ADD      I
SHIFTL   a
ADD      J
ADD      ADDR_A
ADD      _add_op
```

_add_op: ADD 0

ADD (0x0012+2^a*i+j)

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
        CLEAR
        ADD     I
        SHIFTL  a
        ADD     J
        ADD     ADDR_A
        ADD     _add_op
        STORE   _L1

_L1:    CLEAR
```

ADD (0x0012+2^a*i+j)

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
      CLEAR
      ADD      I
      SHIFTL   a
      ADD      J
      ADD      ADDR_A
      ADD      _add_op
      STORE    _L1
      CLEAR
_L1:  ~~CLEAR~~ ADD (0x0012+2^a*i+j)
```

0

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	0	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
      CLEAR
      ADD      I
      SHIFTL   a
      ADD      J
      ADD      ADDR_A
      ADD      _add_op
      STORE    _L1
      CLEAR
_L1:  ~~CLEAR~~ ADD (0x0012+2^a*i+j)
```

A[i][j]

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	A[i][j]	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

Step 1: Load A[i][j]

```
      CLEAR
      ADD      I
      SHIFTL   a
      ADD      J
      ADD      ADDR_A
      ADD      _add_op
      STORE    _L1
      CLEAR
_L1:  ~~CLEAR~~ ADD (0x0012+2^a*i+j)
      STORE    TMP_1
```

A[i][j]

Accumulator

Self Modifying Code Example

Memory

	.....	0x0000
I :	i	0x0008
J :	j	0x000a
	.....	
A :	A[0][0]	0x0012
	A[0][1]	0x0014
	.....	
	A[2^a-1][2^a-2]	
	A[2^a-1][2^a-1]	
	.....	
B :	0	0x0040
ADDR_A :	0x0012	0x0042
TMP_1 :	**A[i][j]**	0x0044

Pseudo-code:

B = A[i][j] + A[j][i]

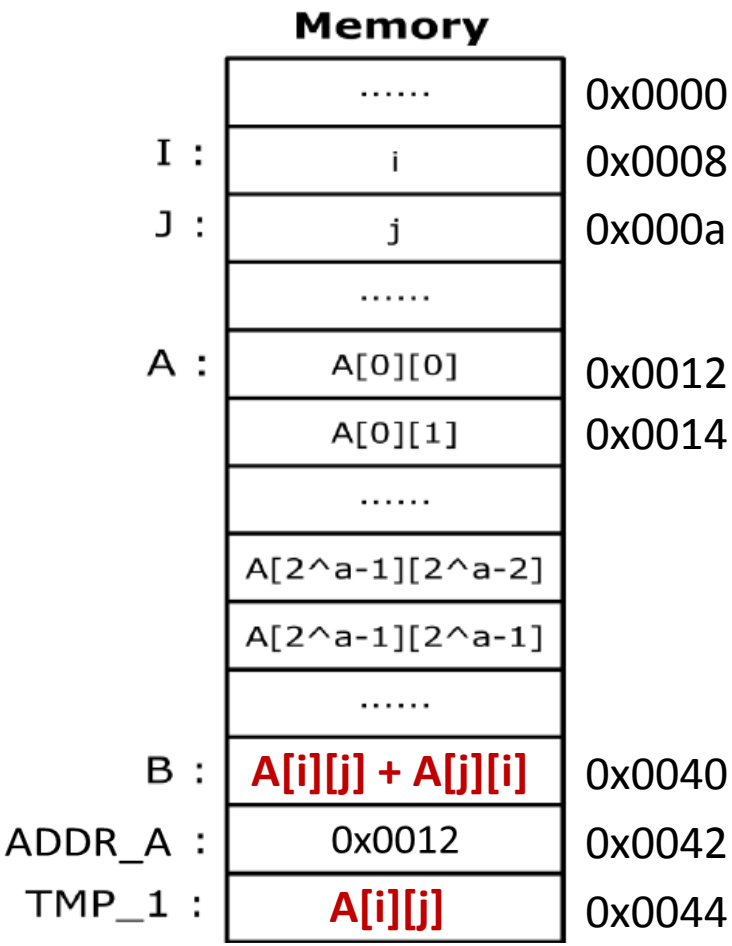
Step 2: Load A[j][i]

```
      CLEAR
      ADD     J
      SHIFTL  a
      ADD     I
      ADD     ADDR_A
      ADD     _add_op
      STORE   _L2
      CLEAR
_L2:  CLEAR
```

A[j][i]

Accumulator

Self Modifying Code Example



Pseudo-code:

B = A[i][j] + A[j][i]

Step 2: Load A[j][i]

```
      CLEAR
      ADD     J
      SHIFTL  a
      ADD     I
      ADD     ADDR_A
      ADD     _add_op
      STORE   _L2
      CLEAR
_L2:  CLEAR
```

Step 3: Store A[i][j] + A[j][i]

```
      ADD     TMP_1
      STORE   B
```