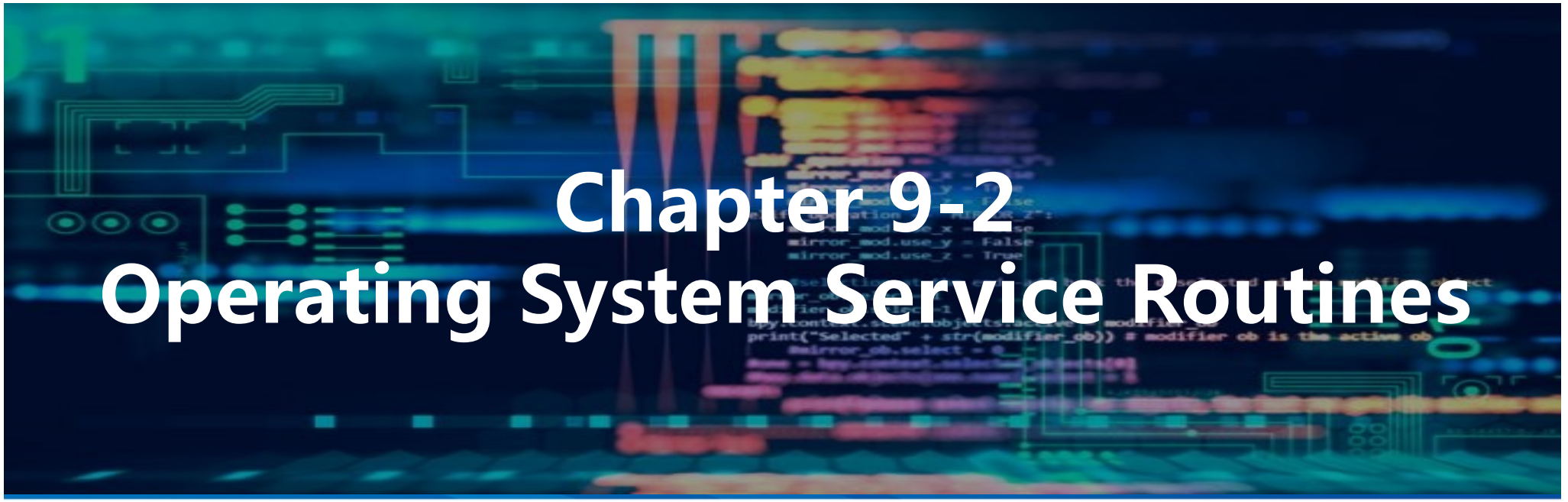
中国科学技术大学
University of Science and Technology of China

计算系统概论A
Introduction to Computing Systems
（ CS1002A.03 ）



Chapter 9-2
Operating System Service Routines

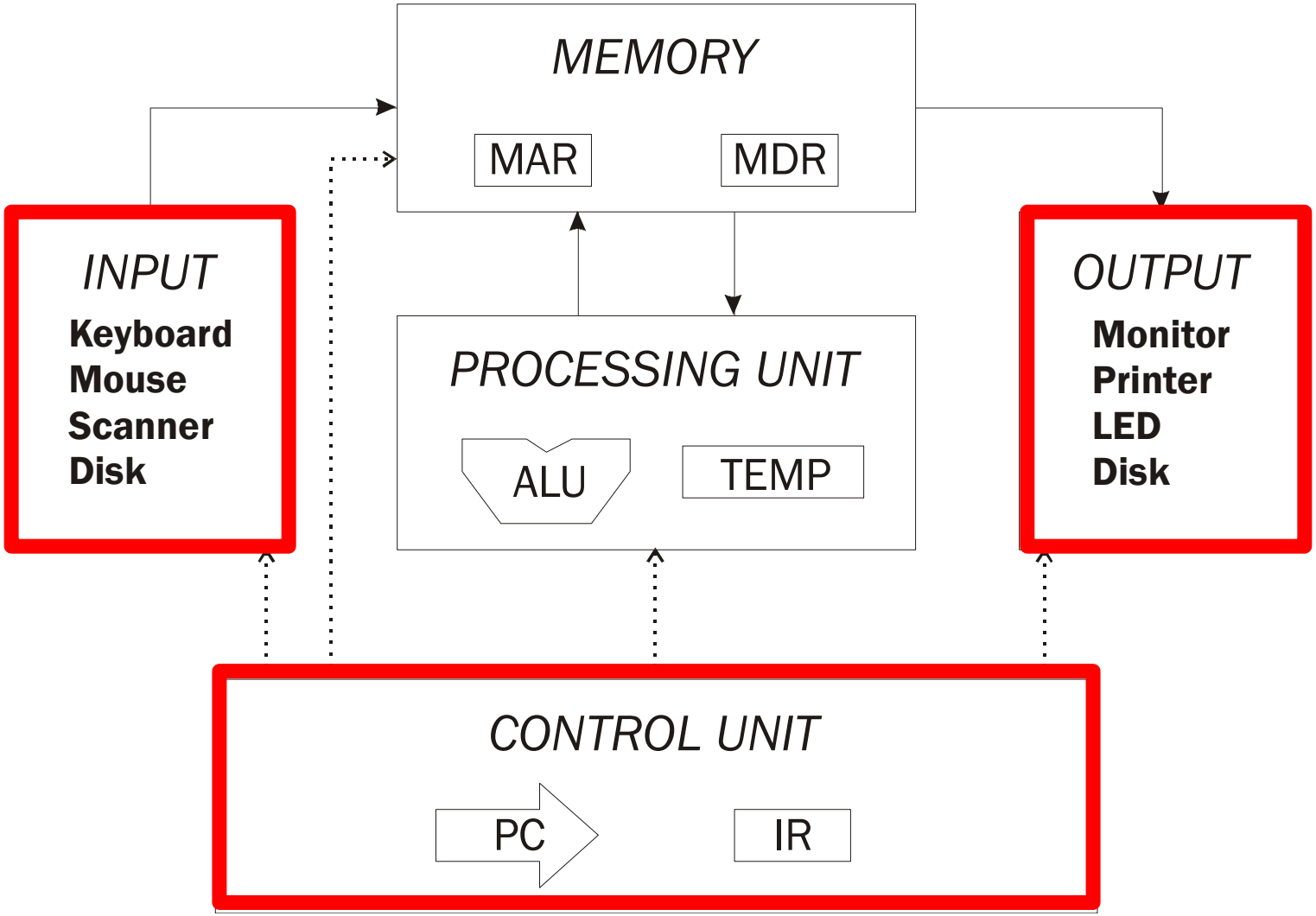
陈俊仕
cjuns@ustc.edu.cn
2023 Fall

计算机科学与技术学院
School of Computer Science and Technology

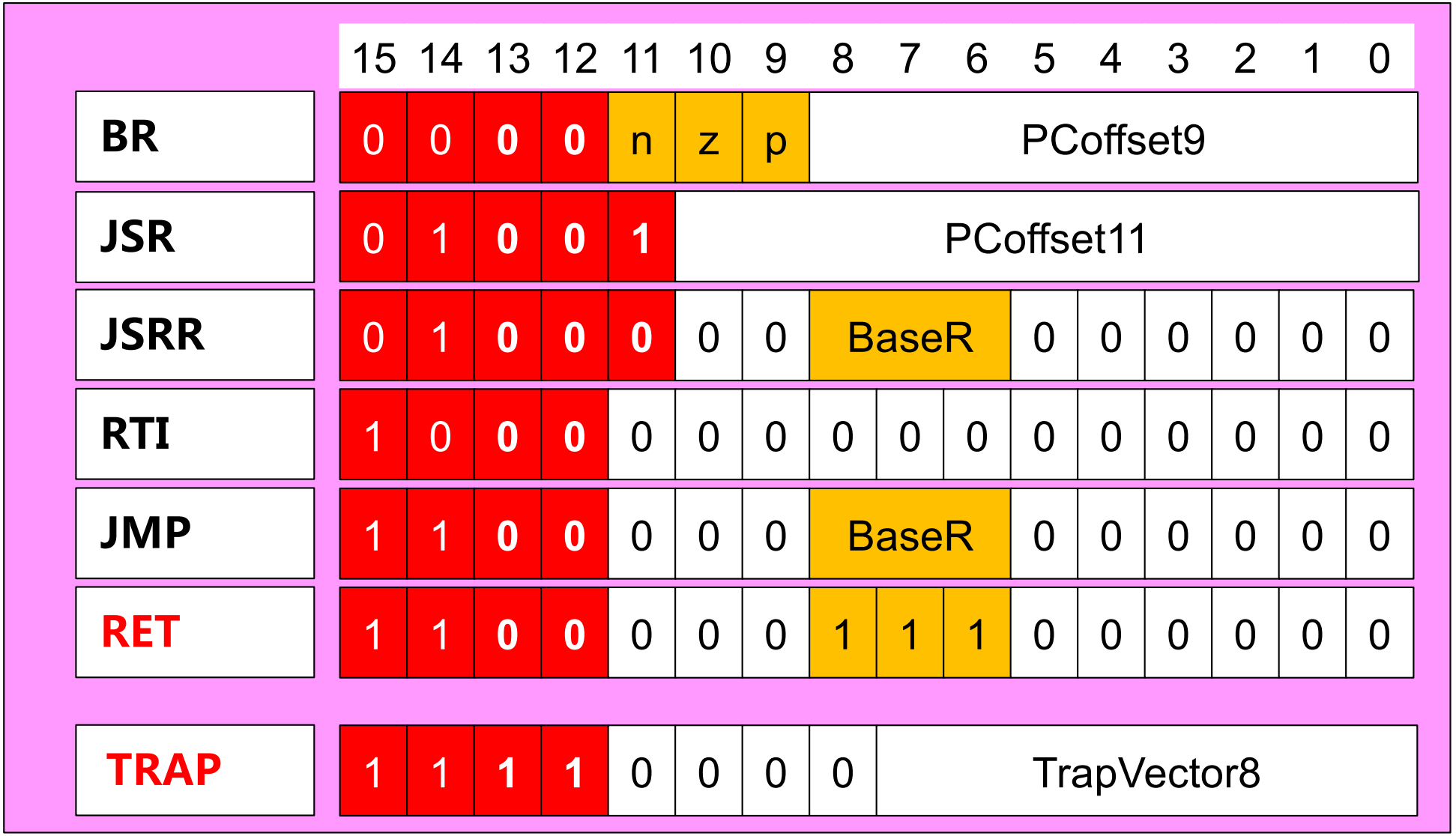
Outline

中国科学技术大学
University of Science and Technology of China

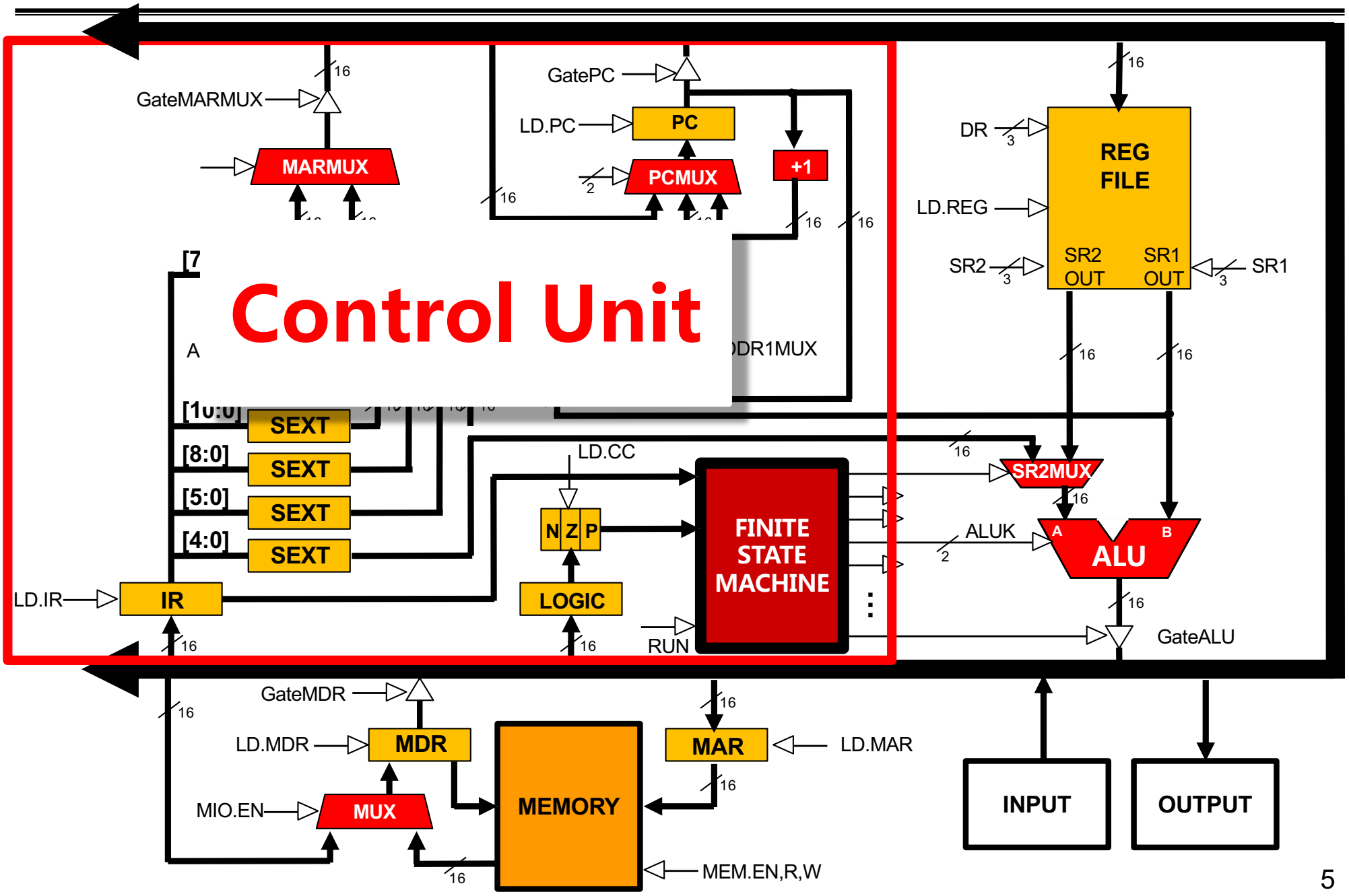
Today: CONTROL in Von Neumann Model



Control Instructions for System Calls



Today: CONTROL in LC-3 Data Path



Outline

中国科学技术大学
University of Science and Technology of China

System Calls

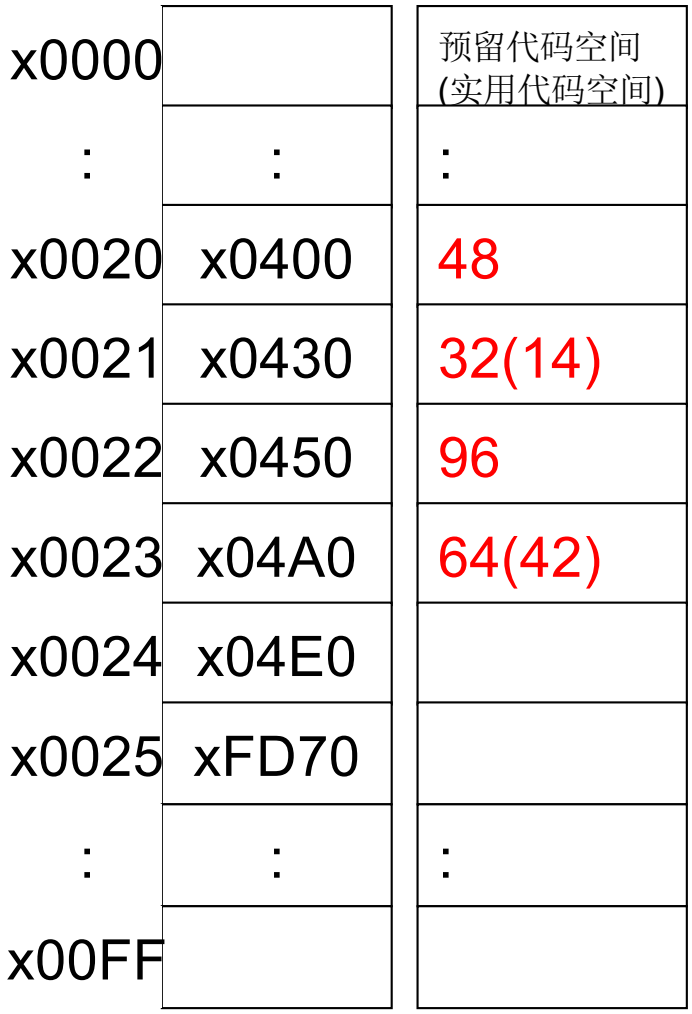
- Some ops. require specialized knowledge and protection
  - Abstract I/O device registers and how to use them, Programmers don't want to know this!
  - Protection for shared I/O resources - isolate programs from OS
  - Reuse of common code
- Solution: *service routines* or *system calls*
  - Low-level, privileged operations performed by operating system

- 1. User program invokes system call
- 2. Operating system code:
  - Saves registers
  - Performs operation
  - Restores registers
- 3. Returns control to user program

LC-3 TRAP Mechanism

- ***Provides set of service routines***
  - **Part of operating system -- routines start at arbitrary addresses**
  - **Up to 256 routines**
- ***Requires table of starting addresses***
  - **Stored in memory (x0000 through x00FF)**
  - **Used to associate code with trap number**
  - **Called System Control Block (or Trap Vector Table)**
- ***Uses TRAP instruction***
  - **Used by program to transfer control to operating system (w/ privileges)**
  - **8-bit trap vector names one of the 256 service routines**
- ***Uses "RTI" instruction***
  - **Returns control to the user program (w/o privileges)**
  - **Execution resumes immediately after the TRAP instruction**

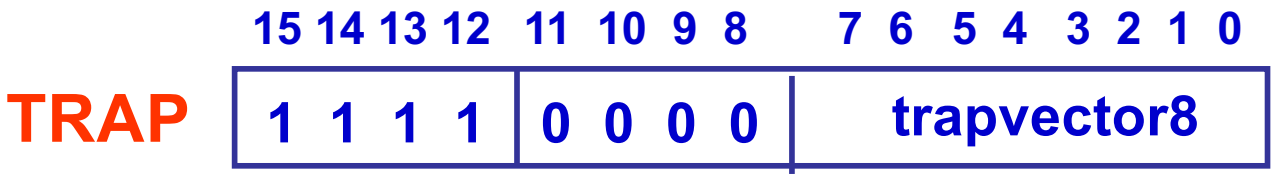
LC-3 TRAP Routines(*service routines*)



The LC-3 System Control Block

- GETC (TRAP x20)
  - Read a single character from KBD.
  - ASCII copied in R0, R0[15:8] cleared.
- OUT (TRAP x21)
  - Write R0[7:0] to CRT.
- PUTS (TRAP x22)
  - Write a string to CRT. String address in R0.
- IN (TRAP x23)
  - Print a prompt on the screen and read a single character from KBD.
  - Character is echoed to Display. ASCII copied to R0[7:0] and R0[15:8] cleared.
- HALT (TRAP x25)
  - Prints message on CRT & halts execution.

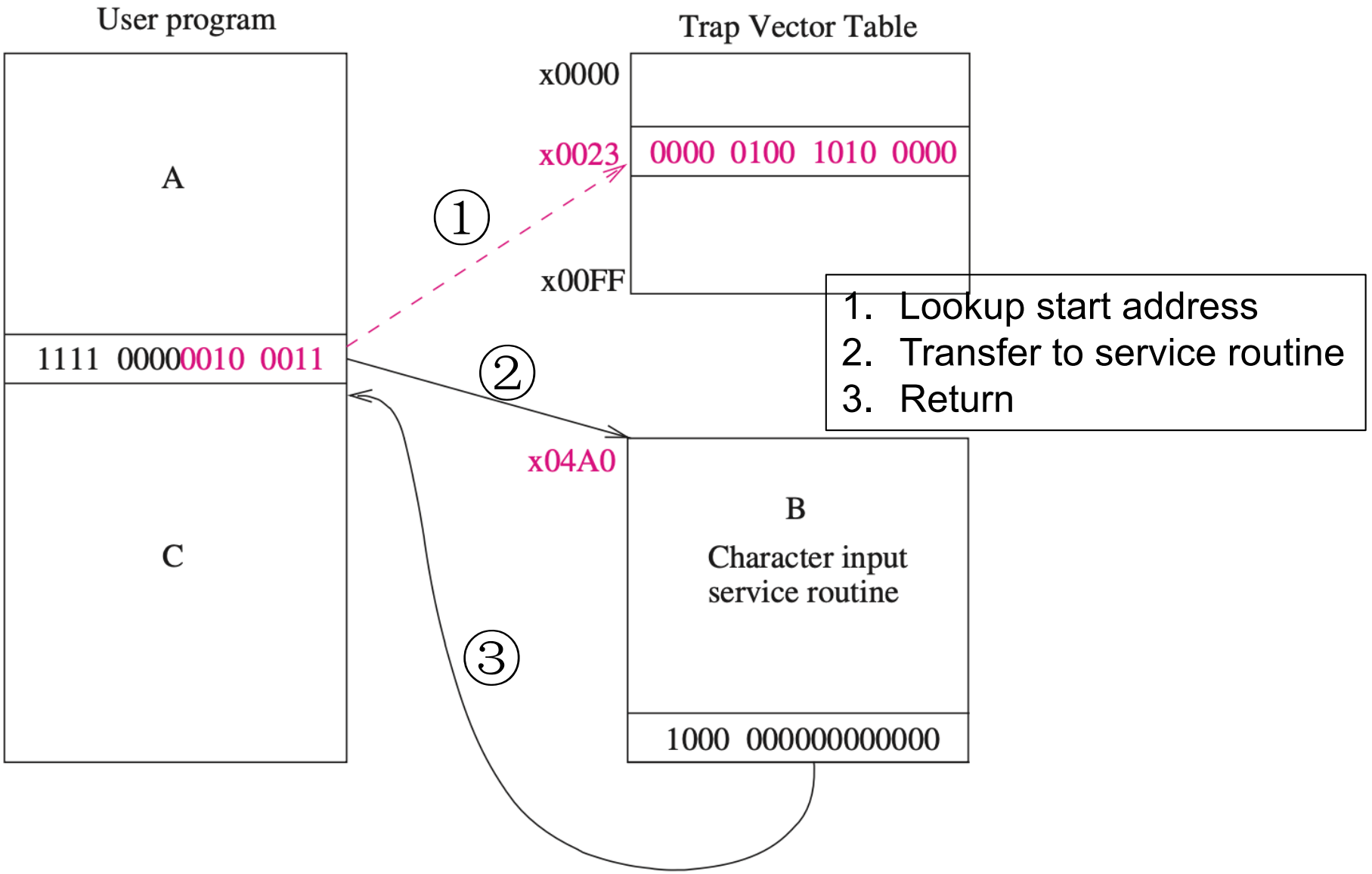
TRAP Instruction



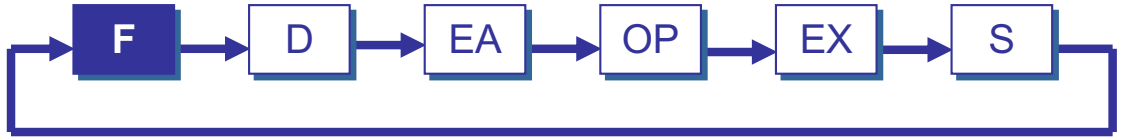
- **Trap vector**
  - Identifies which system call to invoke
  - Serves as index into table of service routine addresses
    - LC-3: table stored in memory at 0x0000 – 0x00FF
    - 8-bit trap vector zero-extended to form 16-bit address
  - Enters privileged mode (PSR[15]<-0)

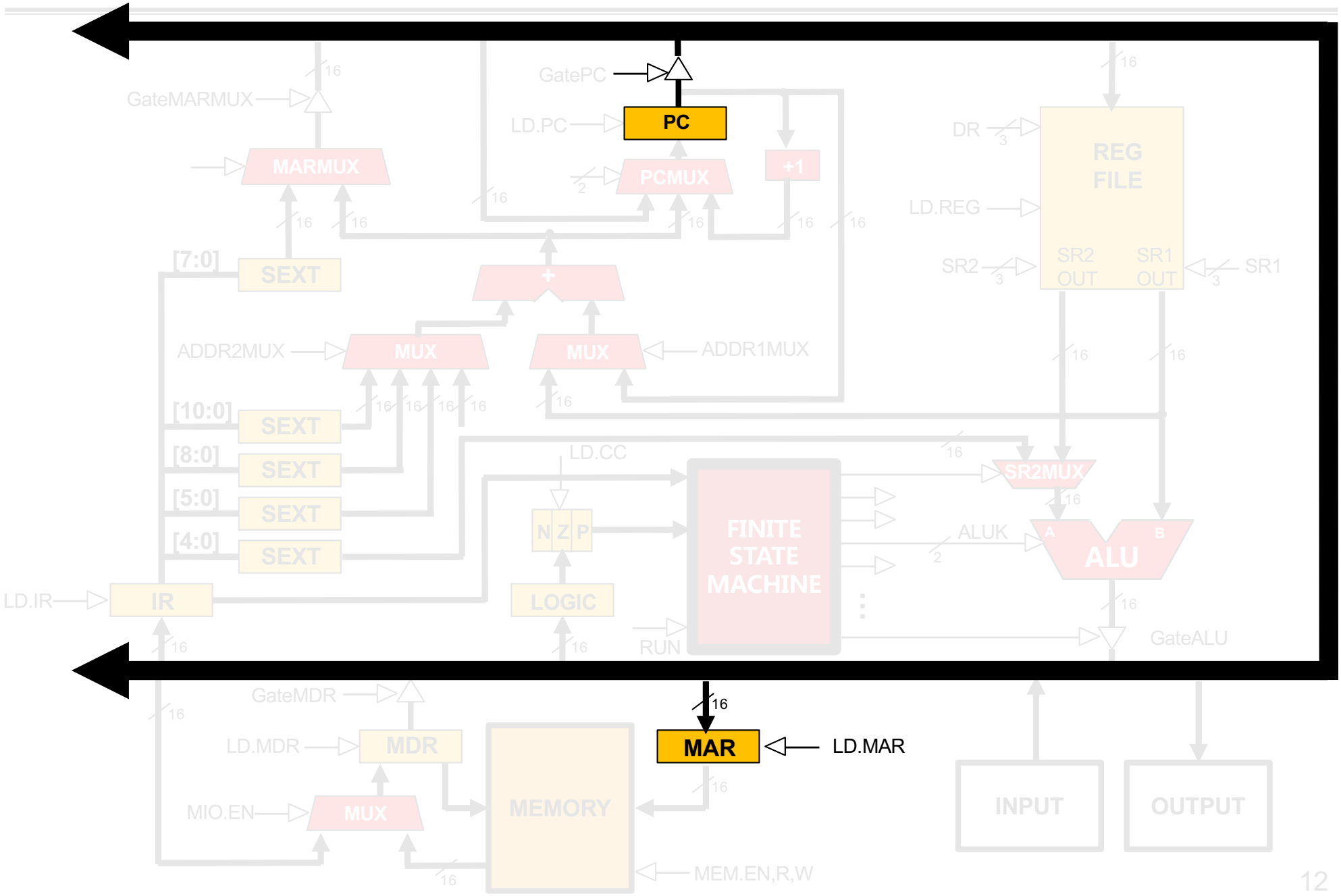
- **Where to go**
  - Lookup starting address from table; place in PC
- **Enabling return**
  - Save PC after TRAP and PSR to system stack
  - Saved_USP <- R6, Saved_SSP <- R6

TRAP Mechanism Operation

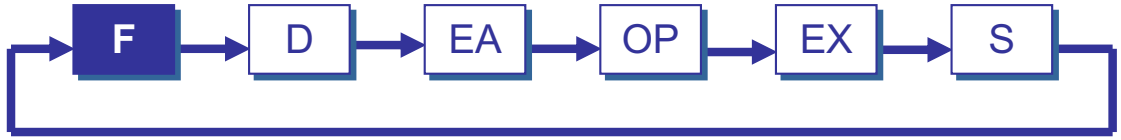


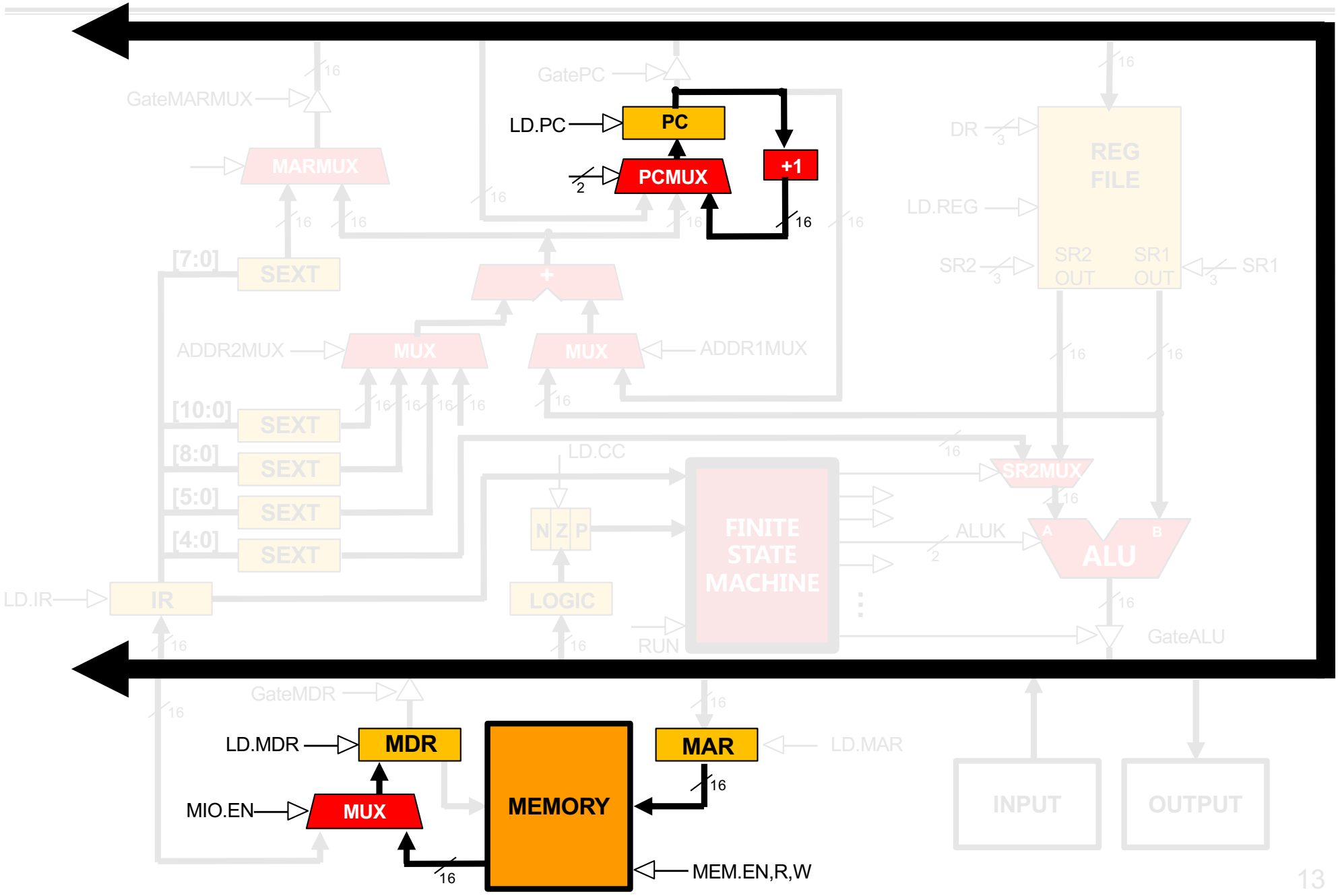
TRAP


F
D
EA
OP
EX
S

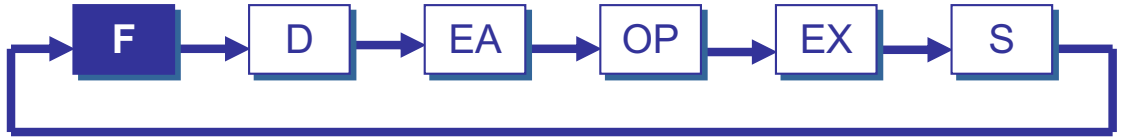

GatePC
LD.PC
PC
GateMARMUX
MARMUX
PCMUX
+1
+
DR
REG FILE
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
[7:0]
SEXT
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.CC
N Z P
FINITE STATE MACHINE
SR2MUX
ALUK
A
B
ALU
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
LD.MDR
MDR
MIO.EN
MUX
MEMORY
MEM.EN,R,W
MAR
LD.MAR
INPUT
OUTPUT

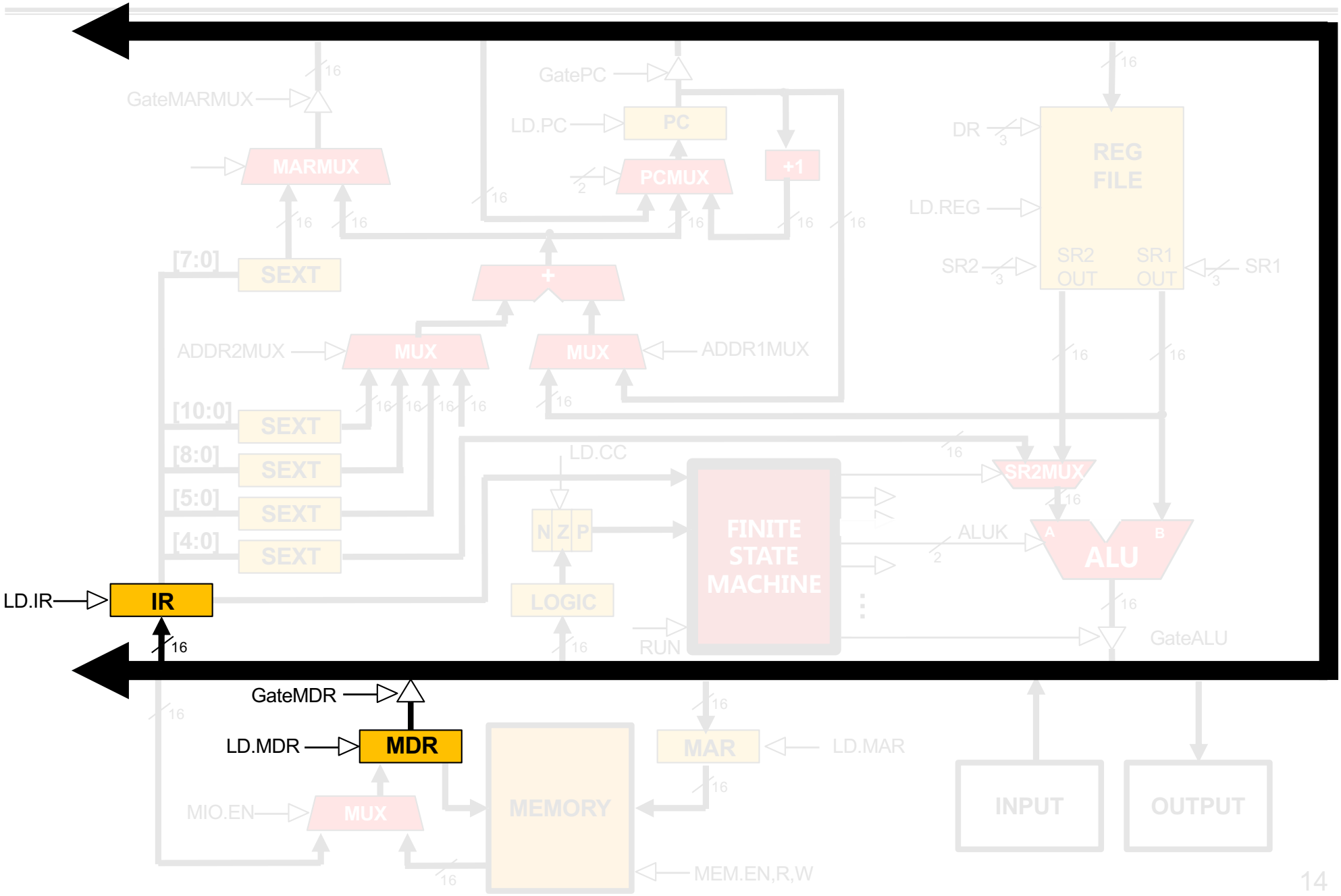
TRAP


F
D
EA
OP
EX
S

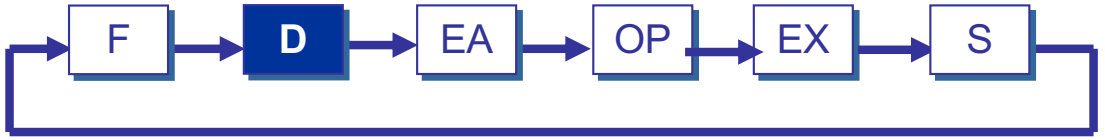

GatePC
LD.PC
PC
PCMUX
+1
2
16
GateMARMUX
MARMUX
[7:0]
SEXT
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
SEXT
[8:0]
SEXT
[5:0]
SEXT
[4:0]
SEXT
LD.CC
N Z P
FINITE STATE MACHINE
RUN
LD.IR
IR
LOGIC
DR
REG FILE
LD.REG
SR2
SR2 OUT
SR1 OUT
SR1
SR2MUX
ALUK
A
B
ALU
GateALU
GateMDR
LD.MDR
MDR
MIO.EN
MUX
MEMORY
MAR
LD.MAR
MEM.EN,R,W
INPUT
OUTPUT

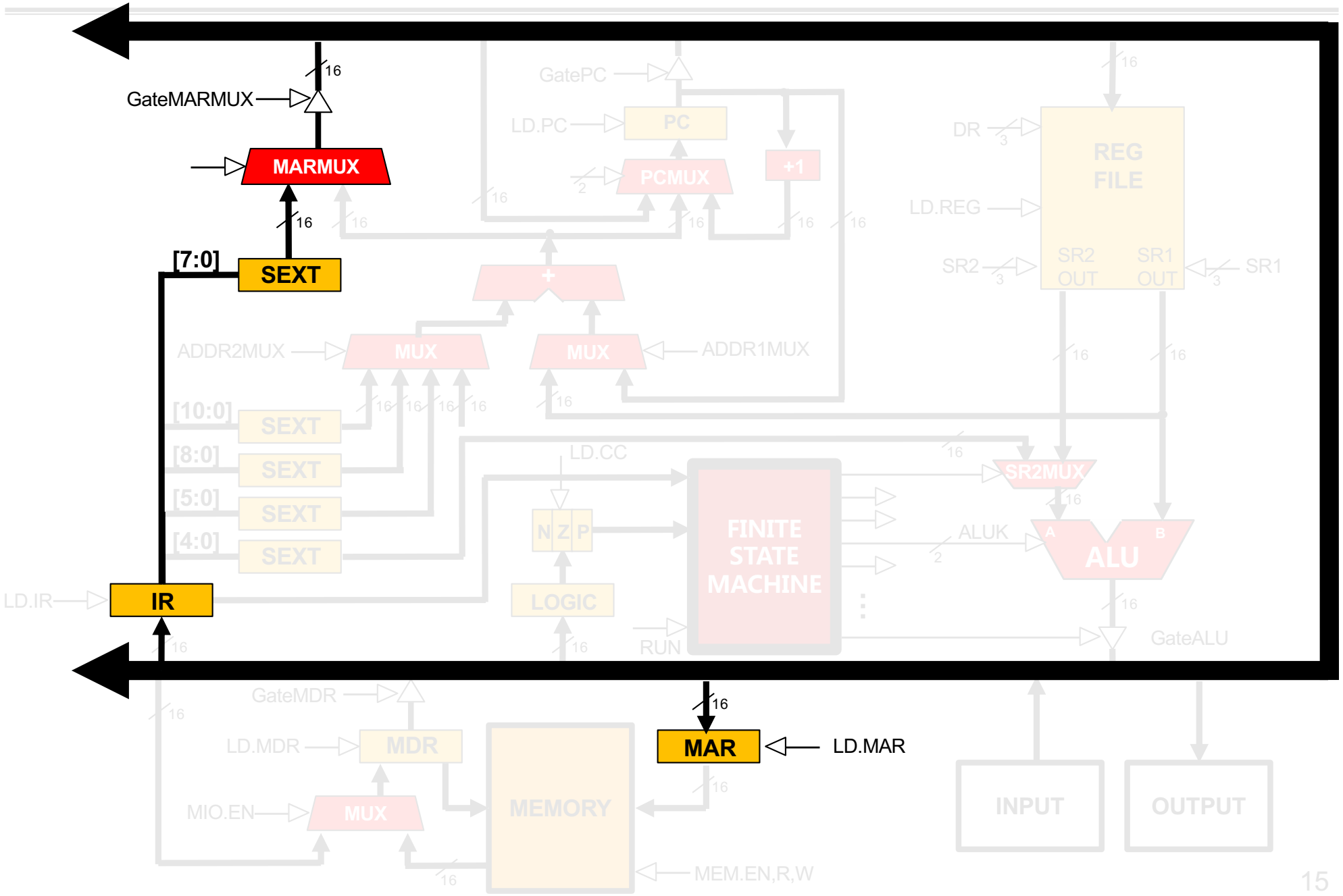
TRAP


F
D
EA
OP
EX
S

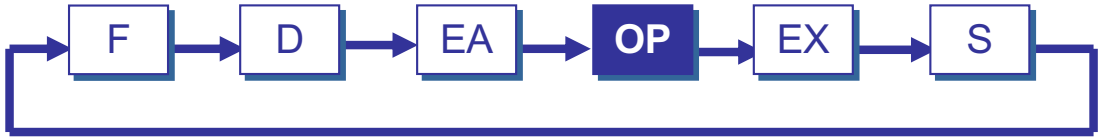

GateMARMUX
MARMUX
GatePC
LD.PC
PC
PCMUX
+1
DR
REG FILE
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
[7:0]
SEXT
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.CC
N Z P
FINITE STATE MACHINE
SR2MUX
ALUK
A
B
ALU
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
LD.MDR
MDR
MAR
LD.MAR
MIO.EN
MUX
MEMORY
MEM.EN,R,W
INPUT
OUTPUT

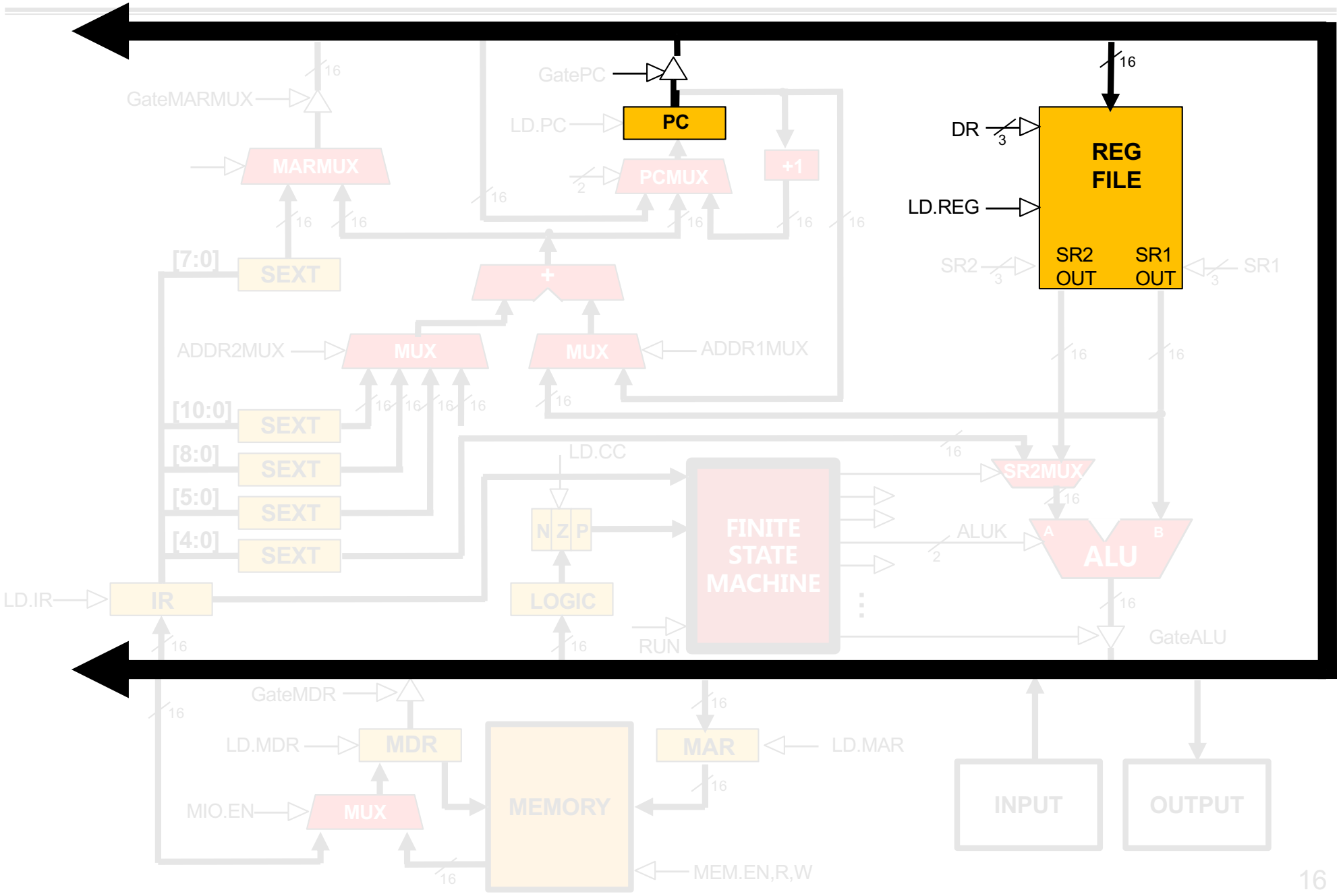
TRAP


F
D
EA
OP
EX
S

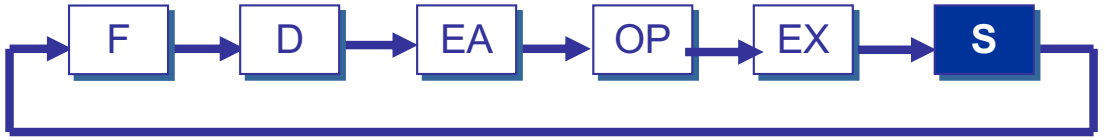

16
GateMARMUX
MARMUX
16
[7:0]
SEXT
GatePC
LD.PC
PC
PCMUX
+1
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
SEXT
[8:0]
SEXT
[5:0]
SEXT
[4:0]
SEXT
LD.CC
N Z P
LOGIC
FINITE STATE MACHINE
RUN
LD.IR
IR
DR
REG FILE
LD.REG
SR2
SR2 OUT
SR1 OUT
SR1
SR2MUX
ALUK
A
B
ALU
GateALU
GateMDR
LD.MDR
MDR
MIO.EN
MUX
MEMORY
MEM.EN,R,W
MAR
LD.MAR
INPUT
OUTPUT

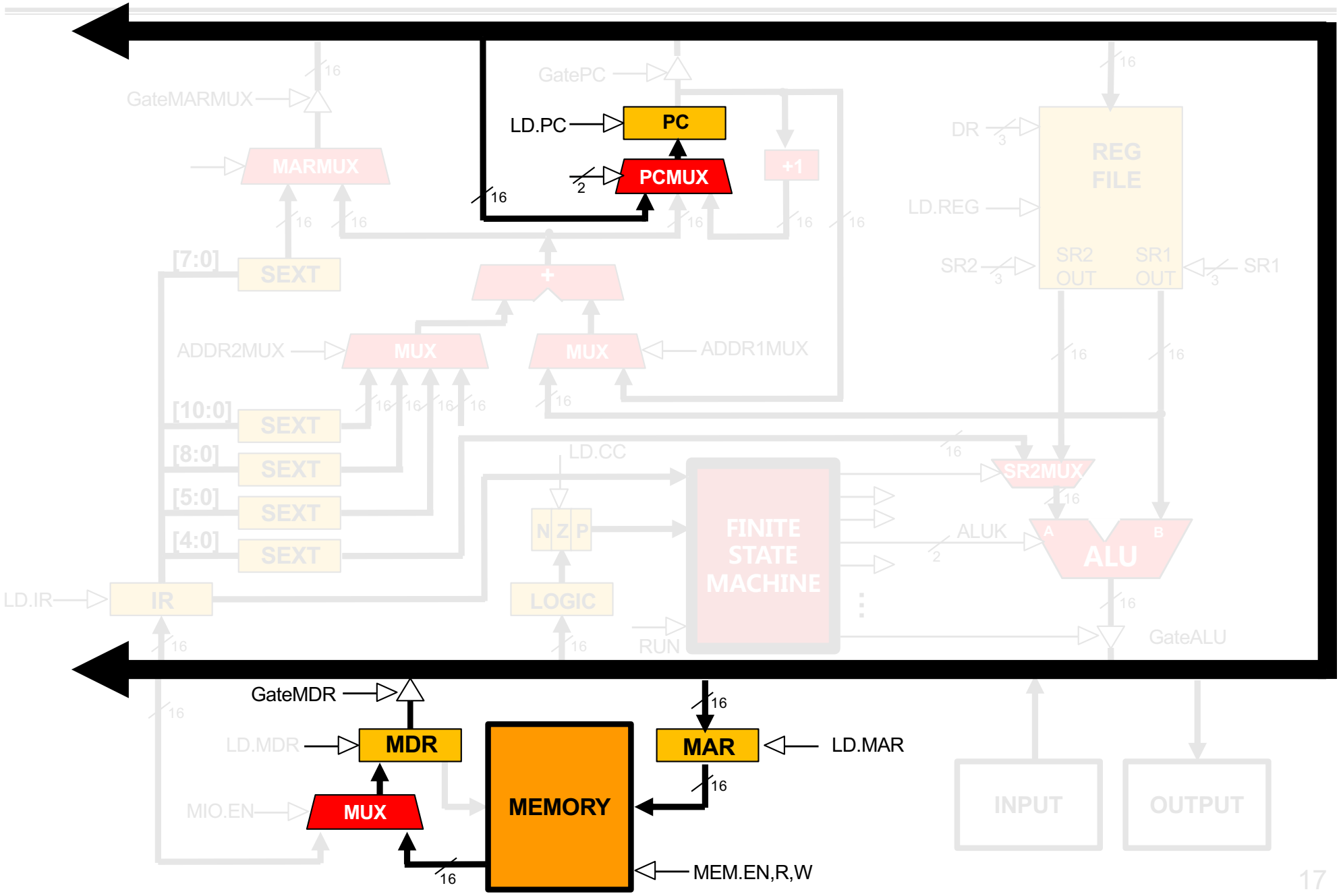
TRAP


F
D
EA
OP
EX
S


GatePC
LD.PC
PC
+1
PCMUX
GateMARMUX
MARMUX
16
DR
3
REG
FILE
LD.REG
SR2
OUT
SR1
OUT
SR2
SR1
[7:0]
SEXT
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.CC
SR2MUX
ALUK
ALU
N Z P
FINITE
STATE
MACHINE
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
LD.MDR
MDR
MAR
LD.MAR
MIO.EN
MUX
MEMORY
MEM.EN,R,W
INPUT
OUTPUT

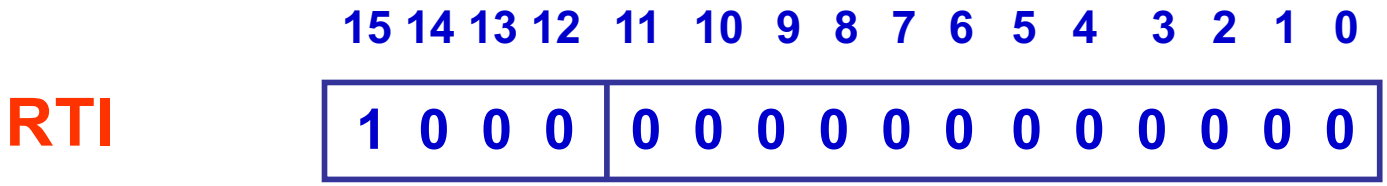
TRAP


F
D
EA
OP
EX
S


GatePC
LD.PC
PC
PCMUX
+1
16
2
GateMARMUX
MARMUX
DR
REG FILE
LD.REG
SR2
SR2 OUT
SR1 OUT
SR1
3
[7:0]
SEXT
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.CC
N Z P
FINITE STATE MACHINE
SR2MUX
ALUK
A
B
ALU
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
MDR
LD.MDR
MIO.EN
MUX
MEMORY
MAR
LD.MAR
MEM.EN,R,W
INPUT
OUTPUT

RTI instruction

- **RTI – Return from Trap or Interrupt instruction**
  - **How to return**
    - **Pop the top two values on the system stack into the PC and PSR.**
    - **Place address in R7 in PC, Return the execution to the last calling point.**
  - **R6 <- Saved_USP(PSR[15]==1) or Saved_SSP(PSR[15]==0)**



Caution Using TRAPs

```
                LEA     R3,BLOCK        ;Init. To first loc.
                LD      R6,ASCII        ;Char->digit template
                LD      R7,COUNT        ;Init. to 10
AGAIN           TRAP    x23             ;Get char
                ADD     R0,R0,R6        ;Convert to number
                STR     R0,R3,#0        ;Store number
                ADD     R3,R3,#1        ;Incr pointer
                ADD     R7,R7,-1        ;Decr counter
                BRp     AGAIN           ;More?
                BRnzp NEXT_TASK
ASCII           .FILL xFFD0   ;Negative of x0023
COUNT           .FILL #10
BLOCK           .BLKW #10
```

What's wrong with this code?

Caution Using TRAPs(*"caller-save" in User code*)

```
                LEA     R3,BLOCK        ;Init. To first loc.
                LD      R6,ASCII        ;Char->digit template
                LD      R7,COUNT        ;Init. to 10
AGAIN           ST      R7,SaveR7
                TRAP    x23             ;Get char
                LD      R7,SaveR7
                ADD     R0,R0,R6        ;Convert to number
                STR     R0,R3,#0        ;Store number
                ADD     R3,R3,#1        ;Incr pointer
                ADD     R7,R7,-1        ;Decr counter
                BRp     AGAIN           ;More?
                BRnzp   NEXT_TASK
SaveR7          .BKLW 1
ASCII           .FILL xFFD0   ;Negative of x0023
COUNT           .FILL #10
BLOCK           .BLKW #10
```

Outline

中国科学技术大学
University of Science and Technology of China

Character Input Service Routine (IN, TRAP x23)

```
01; Service Routine for Keyboard Input
02        .ORIG   x04A0           ;System call starting address
03
04;START ST       R7,SaveR7       ;Save the linkage back to the
05;                               ;program?
06        ST      R1,SaveR1       ;Save the values in the registers
07        ST      R2,SaveR2       ;that are used so that they can
08        ST      R3,SaveR3       ;be restored before RET
09
10;Output Newline on CRT
11        LD      R2,Newline
12 L1     LDI     R3,DSR          ;Check DDR—is it free?
13        BRzp    L1              ;Loop until monitor is ready
14        STI     R2,DDR          ;Move cursor to new clean line
15;
16;Output "Input a character"
17        LEA     R1,Prompt       ;Prompt is starting address
18                                ;of prompt string
19 Loop   LDR     R0,R1,#0        ;Get next prompt character
20        BRzp    Input           ;Check for end of prompt string
21 L2     LDI     R3,DSR
22        BRzp    L2
23        STI     R0,DDR          ;Write next character of prompt
24                                ;string
25        ADD     R1,R1,#1        ;Increment prompt point
26        BRnzp   Loop
```

Character Input Service Routine (IN, TRAP x23)

```
27;Input a character from KB
28 Input LDI     R3,KBSR         ;Has a character been typed?
29       BRzp    Input
30       LDI     R0,KBDR         ;Load it into R0
31
32;Echo the character on CRT
33 L3    LDI     R3,DSR
34       BRzp    L3
35       STI     R0,DDR          ;Echo input character to the
36                               ;monitor
37;Output Newline on CRT
38 L4    LDI     R3,DSR          ;Check CRTDR—is it free?
39       BRzp    L4
40       STI     R2,DDR          ;Move cursor to new clean line
41
42;Restore
43       LD      R1,SaveR1       ;Service routine done, restore
44       LD      R2,SaveR2       ;original values in registers.
45       LD      R3,SaveR3       ;
46;      LD      R7,SaveR7       ;Restore linkage back
47;                               ;prior to RET?
48       RET                     ;Return to calling program
```

Character Input Service Routine (IN, TRAP x23)

```
49;Memory for registers saved
50;      SaveR7  .FILL x0000
51       SaveR1  .FILL x0000
52       SaveR2  .FILL x0000
53       SaveR3  .FILL x0000
54
55       DSR     .FILL xF3FC
56       DDR     .FILL xF3FF
57       KBSR    .FILL xF400
58       KBDR    .FILL xF401
59 ;
59       Newline .FILL x000A    ;ASCII code for newline
60       Prompt  .STRINGZ"Input a character>"
61               .END
```

Character Output Service Routine (OUT, TRAP x21)

```
01        .ORIG   x0430           ;System call starting address
02;       ST      R7,SaveR7       ;Save R7 so we can RET at the
03;                               ;bottom?
04        ST      R1,SaveR1       ;R1 will be used to poll the CRT
06                                ;hardware
07;Write the character
08 TryW   LDI     R1,CRTSR        ;Get status
09        BRzp    TryW            ;Bit 15 on says CRT is ready
10 WIt    STI     R0,CRTDR        ;Write character
11
12;Return from trap
13 Retn   LD      R1,SaveR1       ;Restore registers
14;       LD      R7,SaveR7       ;Restore jump return
15        RET                     ;Return from trap
16
17 CRTSR          .FILL xF3FC     ;Address of CRT status register
18 CRTDR          .FILL xF3FF     ;Address of CRT data register
19 SaveR1         .FILL x0000
20; SaveR7        .FILL x0000
21                .END
```

A String Output Service Routine (PUTS,TRAP x22)

```
01;puts.asm (TRAP x22)
02;R0 points to null-terminated string
03;R0, R1 & R3 saved; R7 is lost.
04              .ORIG  x0450
05;             ST     R7,SaveR7
06              ST     R0,SaveR0
07              ST     R1,SaveR1
08              ST     R3,SaveR3
09;
10 LOOP         LDR    R1,R0,#0
11              BRz    RETURN
12 L2           LDI    R3,DSR
13              BRzp   L2
14              STI    R1,DDR
15              ADD    R0,R0,#1
16              BR     LOOP
```

A String Output Service Routine (PUTS,TRAP x22)

```
17 RETURN          LD  R3,SaveR3
18                 LD  R1,SaveR1
19                 LD  R0,SaveR0
20;                LD  R7,SaveR7
21                 RET
22;
23 CRTSR           .FILL     xF3FC
24 CRTDR           .FILL     xF3FF
25 SaveR0          .FILL     x0000
26 SaveR1          .FILL     x0000
27 SaveR3          .FILL     x0000
28; SaveR7         .FILL     x0000
29                 .END
```

Halt the machine(HALT,TRAP x25)

```
01;
02;Service Routine for halting the machine
03;
04        .ORIG   xFD70           ;Where this routine resides
05        ST      R7,SaveR7       ;Save the linkage back to the
06                                ;program?
07        ST      R1,SaveR1       ;R1:a temp for MC register
08        ST      R0,SaveR0       ;R0 is used as working space
09;
10;Print message that machine is halting
11        LD      R0,ASCIINewLine
12        TRAP    x21
13        LEA     R0,Message
14        TRAP    x22             ;Write a string(Message)to CRT
15        LD      R0,ASCIINewLine
16        TRAP    x21
17;
18;Clear bit 15 at xFFFF to stop the machine
19        LDI     R1,MCR          ;Load MC register into R1
20        LD      R0,MASK         ;R0 = x7FFF
21        AND     R0,R1,R0        ;Mask to clear the top bit
22        STI     R0,MCR          ;Store R0 into MC register
```

Halt the machine(HALT,TRAP x25)

```
23;
24;Return from HALT routine
25;{How can this routine return if the machine is halted above?}
26        LD      R1,SaveR1       ;Restore registers
27        LD      R0,SaveR0
28        LD      R7,SaveR7       ;Restore trap return
29        RET
30;
31;Some constants
32 ASCIINewLine .FILL x000A     ;ASCII code for newline
33 SaveR0       .FILL x0000
34 SaveR1       .FILL x0000
35 SaveR7       .FILL x0000
36;
37 Message      .STRING "Halting the machine."
38 MCR          .FILL xFFFE     ;Address of MCR
39 MASK         .FILL x7FFF     ;Mask to clear the top bit
40              .END
```

Data Type Conversion

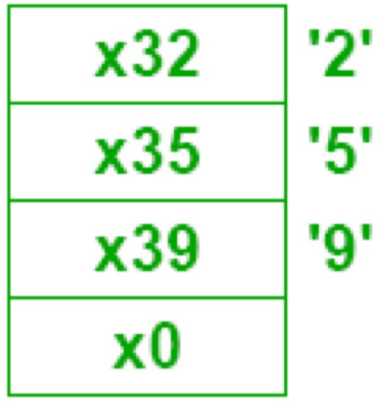
- **I/O**
  - Keyboard input routines read ASCII characters (not binary values)
  - Console output routines write ASCII (‘s’ not “x73”)
- **Consider this program:**

```
TRAP   x23          ; input from keyboard
ADD    R1, R0, #0   ; move to R1
TRAP   x23          ; input from keyboard
ADD    R0, R1, R0   ; add two inputs
TRAP   x21          ; display result
TRAP   x25          ; HALT
```

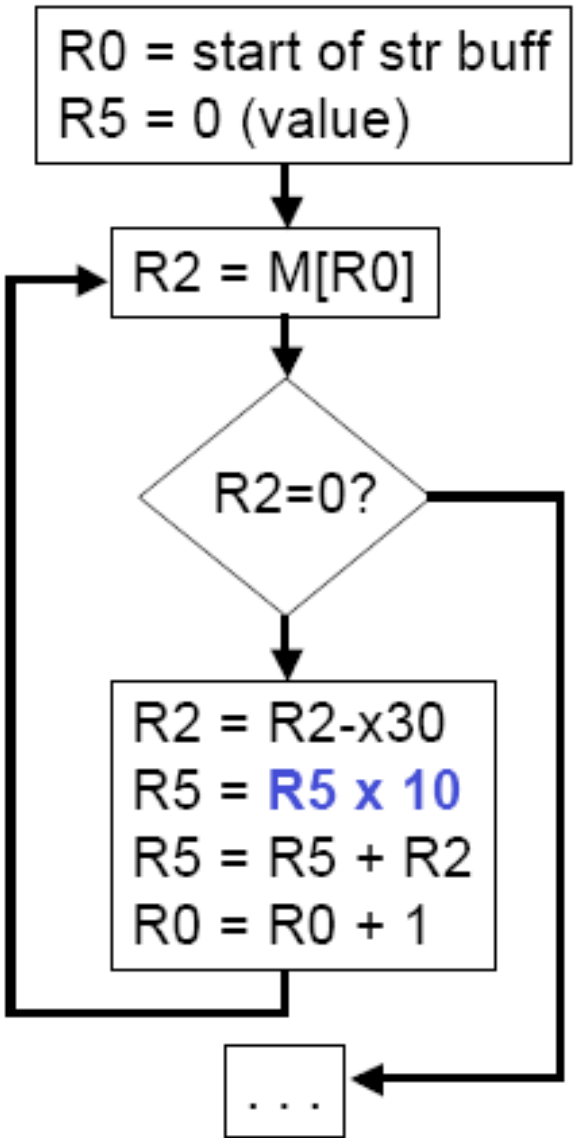
- **User inputs ‘2’ and ‘3’ -- what happens?**
- **Result displayed: ‘e’**
- **Why?**
  - ASCII '2' (x32) + ASCII '3' (x33) = ASCII 'e' (x65)

ASCII to Binary

- **Single digit numbers are trivial** (subtract x30)
  - E.g., '7' is ASCII x37, x37 - x30 = x7
- **Input**
  - **Assume we've read three ASCII digits (*e.g.*, "259") into a memory buffer**
- **How do we convert this to a *number* we ca use?**
  - **Convert first character to digit (subtract x30) and multiply by 100**
  - **Convert second character to digit and multiply by 10**
  - **Convert third character to digit**
  - **Add the three digits together**



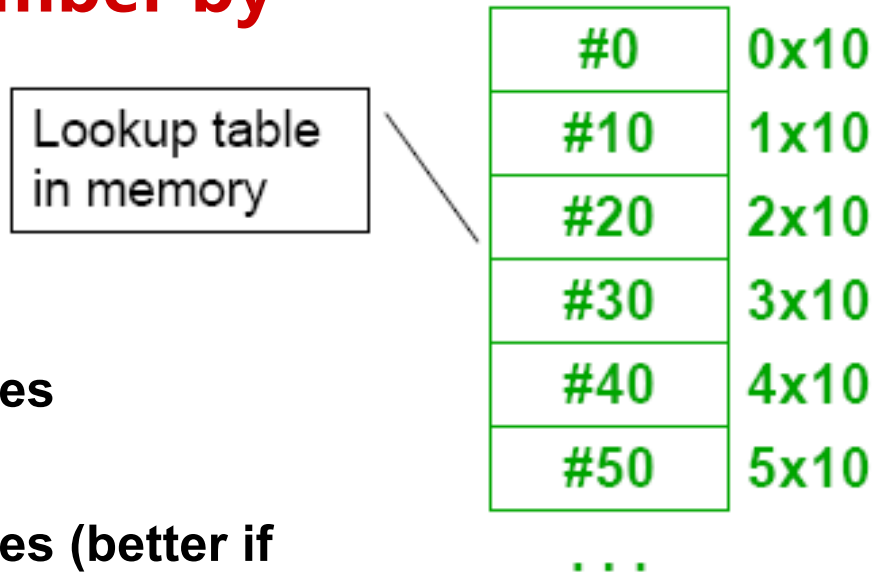
ASCII to Binary Conversion Algorithm



R0 pointer in string buffer
R2 character from memory
R5 current value

Multiplication

- **How can we multiply a number by 100?**
  - `Approach 0`
    - Use the MUL instruction
  - `Approach 1`
    - Add <number> to itself 10 times
  - `Approach 2`
    - Add 10 to itself <number> times (better if number < 10)
  - `Approach 3`
    - Look it up! Only practical if number of multiplicands is small

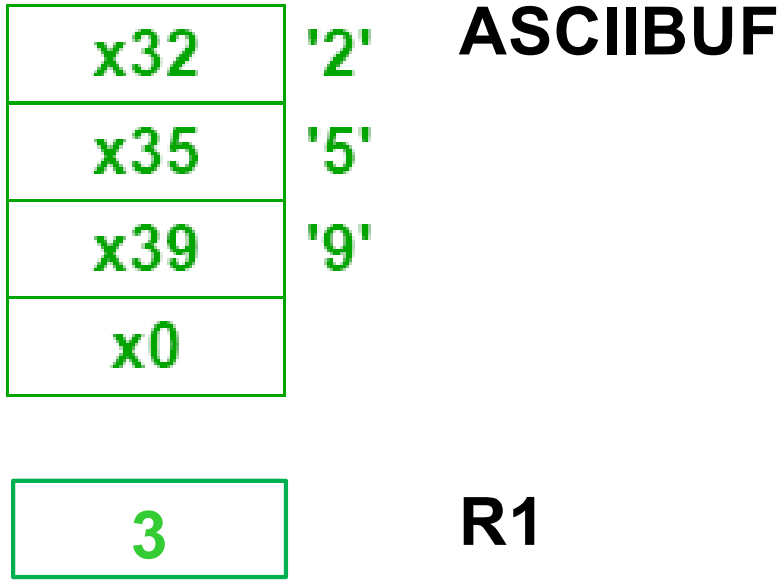


Code for Lookup Table

```
; multiply R0 by 100, using lookup table
;
              LEA  R1, Lookup100   ; R1 = table base
              ADD  R1, R1, R0      ; add index (R0)
              LDR  R0, R1, #0      ; load from M[R1]
              ...
  Lookup100   .FILL #0             ; entry 0
              .FILL #100           ; entry 1
              .FILL #200           ; entry 2
              .FILL #300           ; entry 3
              .FILL #400           ; entry 4
              .FILL #500           ; entry 5
              .FILL #600           ; entry 6
              .FILL #700           ; entry 7
              .FILL #800           ; entry 8
              .FILL #900           ; entry 9
```

Complete ASCII to Binary Conversion Code

- **ASCII "259" to value 259**



Complete ASCII to Binary Conversion Code (1 of 3)

```
; Three-digit buffer at ASCIIBUF.
; R1 tells how many digits to convert.
; Put resulting decimal number in R0.
ASCIItoBinary   AND  R0, R0, #0    ;clear result
                ADD  R1, R1, #0    ;test # digits
                BRz  DoneAtoB      ;done if no digits
;
                LD   R3, NegZero   ;R3 = -x30
                LEA  R2, ASCIIBUF ;ptr to the first digit
                ADD  R2, R2, R1    ;R2 =(R2)+(R1)
                ADD  R2, R2, #-1   ;points to ones digit
;
                LDR  R4, R2, #0    ;load digit
                ADD  R4, R4, R3    ;convert to number
                ADD  R0, R0, R4    ;add ones contribution
;
```

Complete ASCII to Binary Conversion Code(2 of 3)

```
        ADD   R1, R1, #-1     ;one less digit
        BRz   DoneAtoB        ;done if zero
        ADD   R2, R2, #-1     ;points to tens digit
;
        LDR   R4, R2, #0      ;load 'tens' digit
        ADD   R4, R4, R3      ;convert to number
        LEA   R5, Lookup10    ;multiply by 10
        ADD   R5, R5, R4
        LDR   R4, R5, #0
        ADD   R0, R0, R4      ;adds tens contribution to total
;
        ADD   R1, R1, #-1     ;one less digit
        BRz   DoneAtoB        ;done if zero
        ADD   R2, R2, #-1     ;points to hundreds digit
;
        LDR   R4, R2, #0      ;load digit
        ADD   R4, R4, R3      ;convert to number
        LEA   R5, Lookup100   ;multiply by 100
        ADD   R5, R5, R4
        LDR   R4, R5, #0
        ADD   R0, R0, R4      ;adds 100's contrib
```

Complete ASCII to Binary Conversion Code(3 of 3)

```
DoneAtoB        RET
NegZero         .FILL xFFD0   ;-x30
ASCIIBUF        .BLKW 4
Lookup10        .FILL #0
                .FILL #10
                .FILL #20
                ...
Lookup100       .FILL #0
                .FILL #100
                .FILL #200
                ...
```

Binary to ASCII Conversion

- **Converting a 2's complement binary value to a three-digit decimal number**
  - `Resulting characters can be output using OUT`
- **Instead of multiplying, we need to divide by 100 to get hundreds digit.**
  - `Why wouldn't we use a lookup table for this problem?`
  - `Subtract 100 repeatedly from number to divide.`
- **First, check whether number is negative.**
  - `Write sign character (+ or -) to buffer and make positive.`

Binary to ASCII Conversion Code (1 of 3)

```
; R0 is between -999 and +999.
; Put sign character in ASCIIBUF, followed by three
; ASCII digit characters.
BinaryToASCII  LEA R1, ASCIIBUF    ;ptr to result string
               ADD R0, R0, #0      ;test sign of value
               BRn NegSign
               LD  R2, ASCIIplus   ;store '+'
               STR R2, R1, #0
               BR  Begin100
NegSign        LD  R2, ASCIIneg    ;store '-'
               STR R2, R1, #0
               NOT R0, R0          ;convert value to pos
               ADD R0, R0, #1
Begin100       LD  R2, ASCIIoffset
               LD  R3, Neg100
Loop100        ADD R0, R0, R3
               BRn End100
               ADD R2, R2, #1      ;add one to digit
               BR  Loop100
```

Binary to ASCII Conversion Code(2 of 3)

```
End100          STR R2, R1, #1   ;store ASCII 100's digit
                LD  R3, Pos100
                ADD R0, R0, R3   ;restore last subtract
;
                LD  R2, ASCIIoffset
                LD  R3, Neg10
Loop100         ADD R0, R0, R3
                BRn End10
                ADD R2, R2, #1   ;add one to digit
                BR  Loop10
End10           STR R2, R1, #2       ;store ASCII 10's digit
                ADD R0, R0, #10      ;restore last subtract
;
                LD  R2, ASCIIoffset
                ADD R2, R2, R0       ;convert one's digit
                STR R2, R1, #3       ;store one's digit
                RET
;
```

Binary to ASCII Conversion Code(3 of 3)

```
ASCIIplus   .FILL x002B    ;plus sign ASCII code
ASCIIneg    .FILL x002D    ;neg sign ASCII code
ASCIIoffset .FILL x0030    ;zero's ASCII code
Neg100      .FILL xFF9C    ;-100
Pos100      .FILL x0064    ;100
Neg10       .FILL xFFF6    ;-10
```