

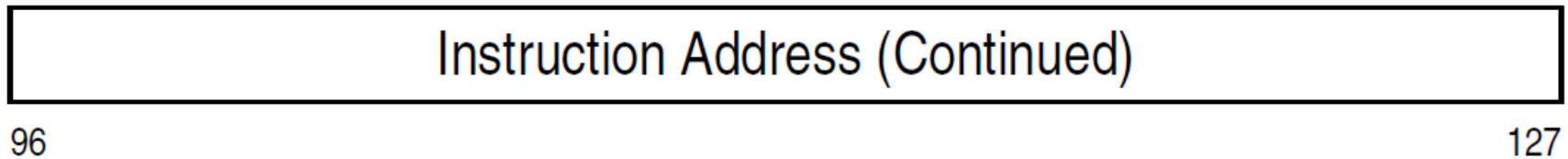
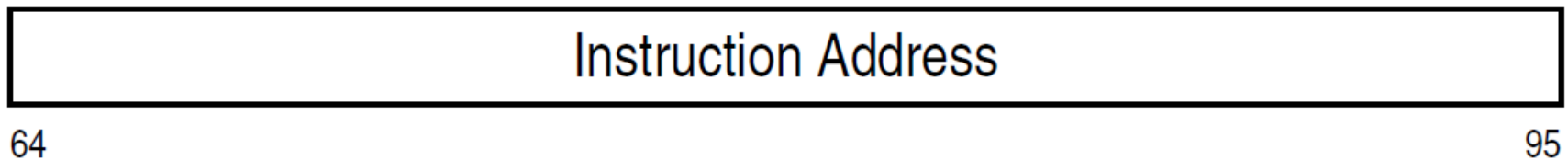
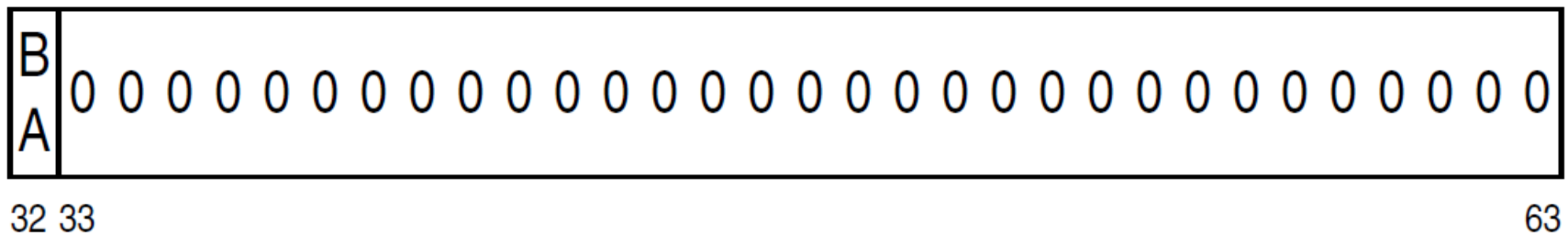
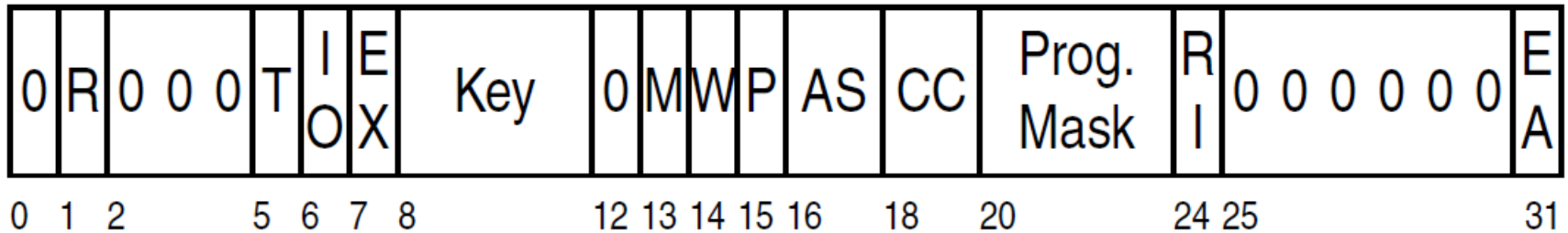
# SHARE Anaheim 2014

## High Level Assembler Bootcamp

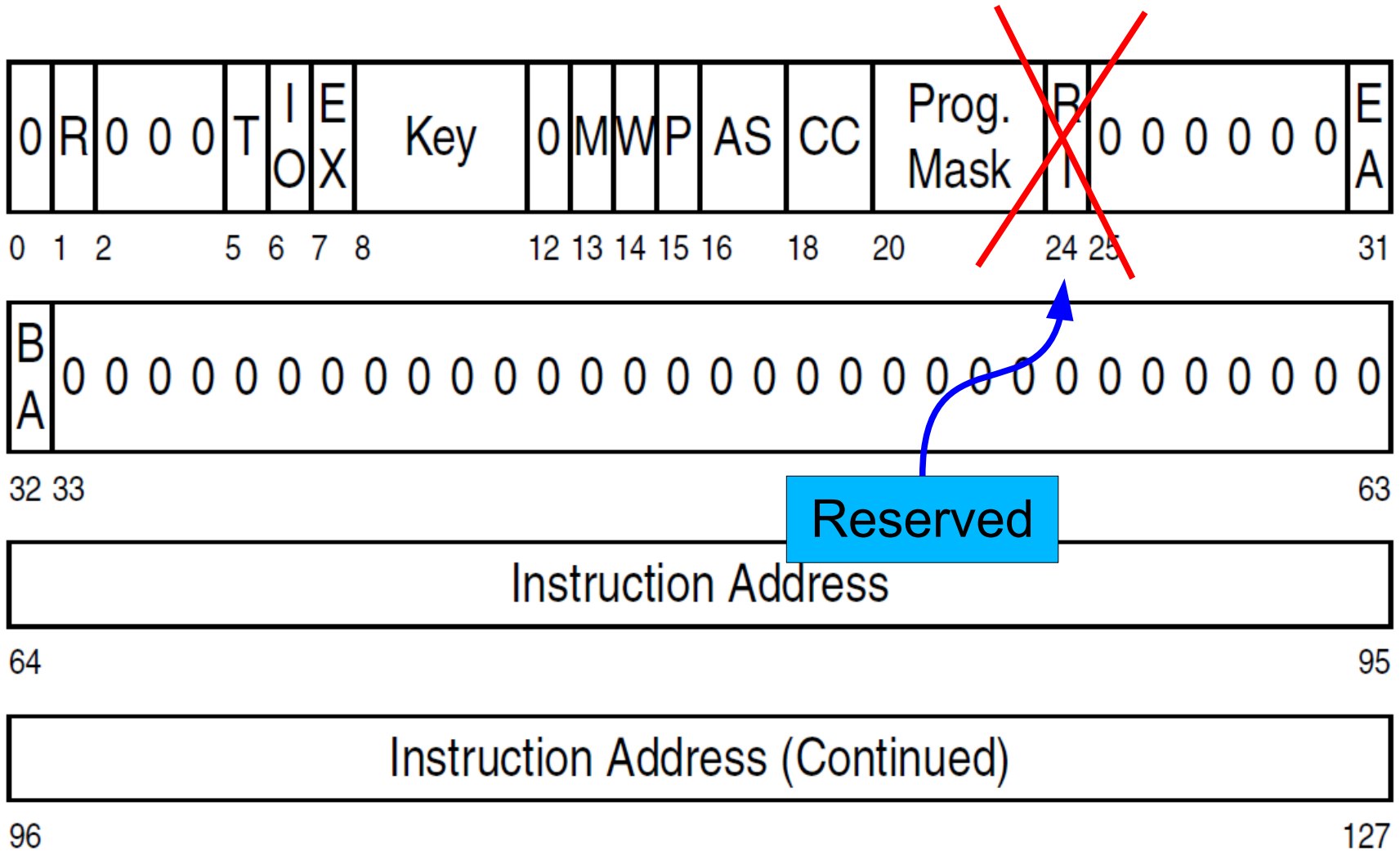
### The Program Status Word (PSW) (15250, 15251, 15252)



## Program Status Word (PSW)



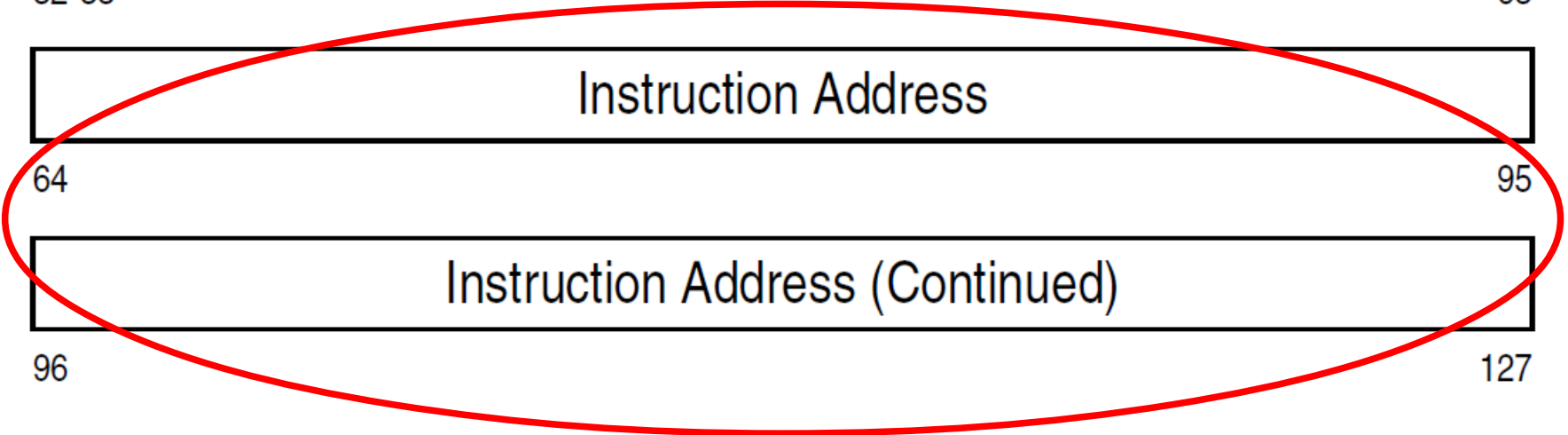
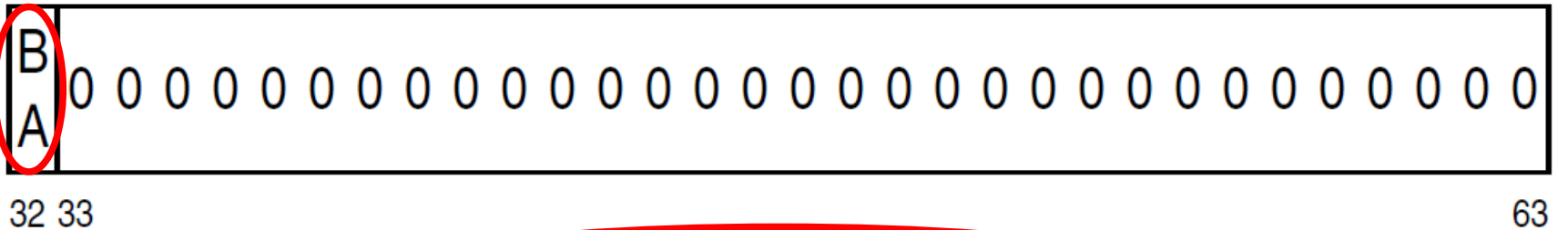
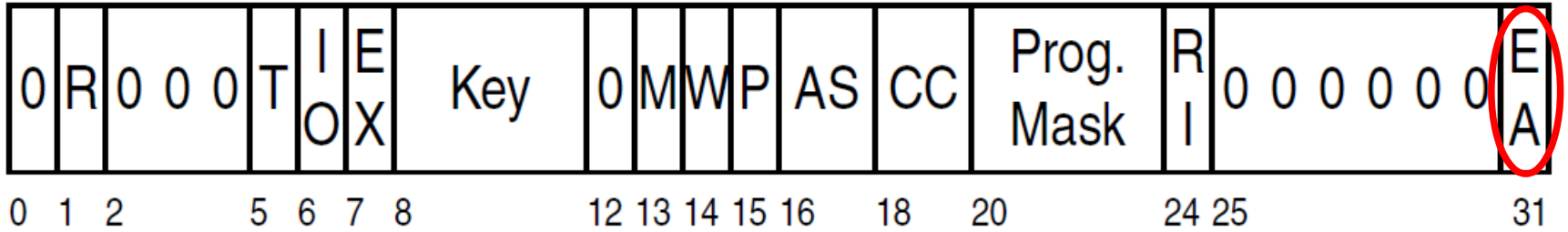
# Program Status Word (PSW)



## Program Status Word (PSW)

- The Program Status Word (PSW) is a register in the processor which includes control information to determine the state of the CPU.
- The z/Architecture PSW is 128-bits in length
  - Bits 0-32 contain flag bits indicating control information for the CPU
  - Bits 33-63 are 0
  - Bits 64-127 contain the instruction address
- EPSW – Extract PSW
  - Obtain bits 0-63 of the PSW and place them into operands of the instruction
- LPSW(E) – Load PSW (Extended)
  - Replace the entire PSW with the contents of storage
  - This means that the instruction branches – well might do...

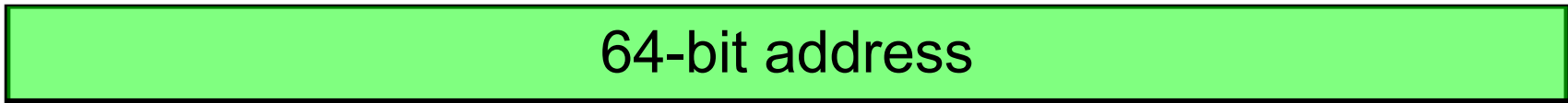
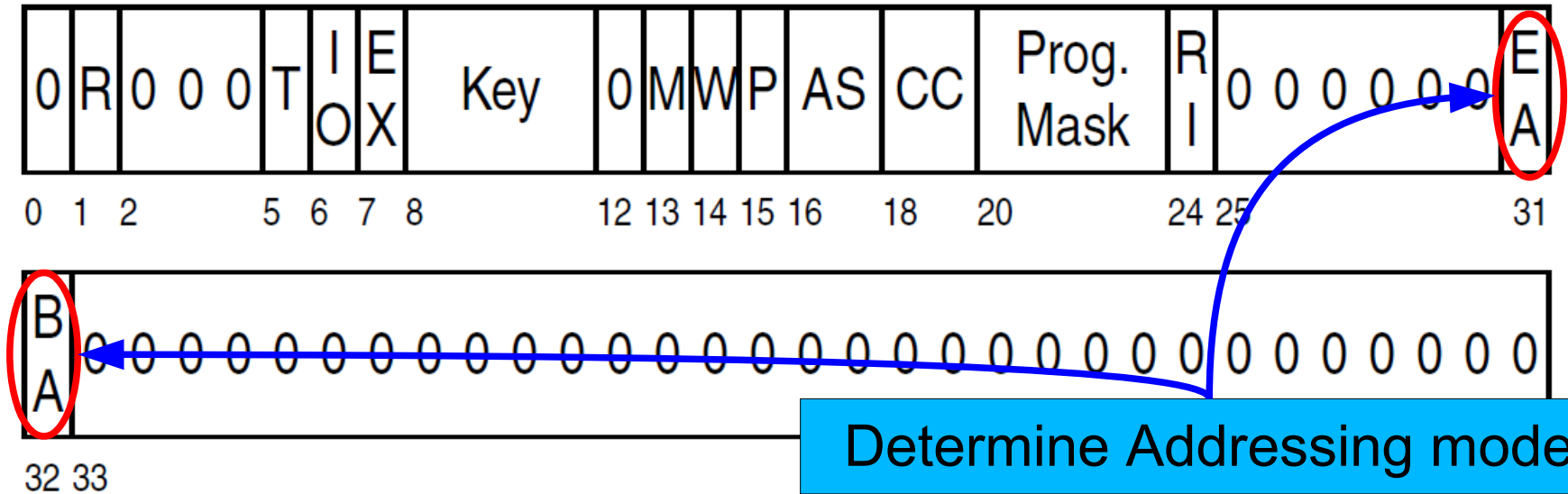
# Program Status Word (PSW) – Addresses



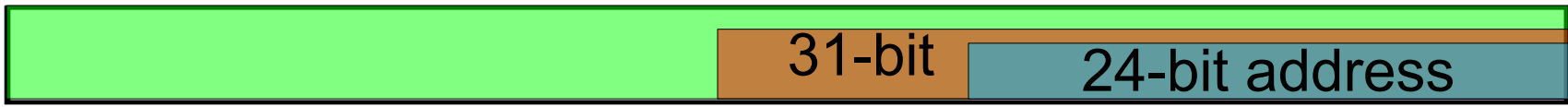
## Program Status Word (PSW) – Addresses

- Since the z/Architecture can run in a number of addressing modes, the instruction address is determined by a variable number of bits in the PSW. The current addressing mode is determined by bits 31-32 of the PSW with the following combinations:
  - 00 → 24-bit mode
  - 01 → 31-bit mode
  - 10 → invalid
  - 11 → 64-bit mode
  
- Bits 64-127 are used to determine the address of the next instruction to be executed
  - However, some instructions may be interrupted and therefore the PSW may point at the same instruction which was being executed so that it is redriven

## Program Status Word (PSW) – Instruction Address



64 95



96 127



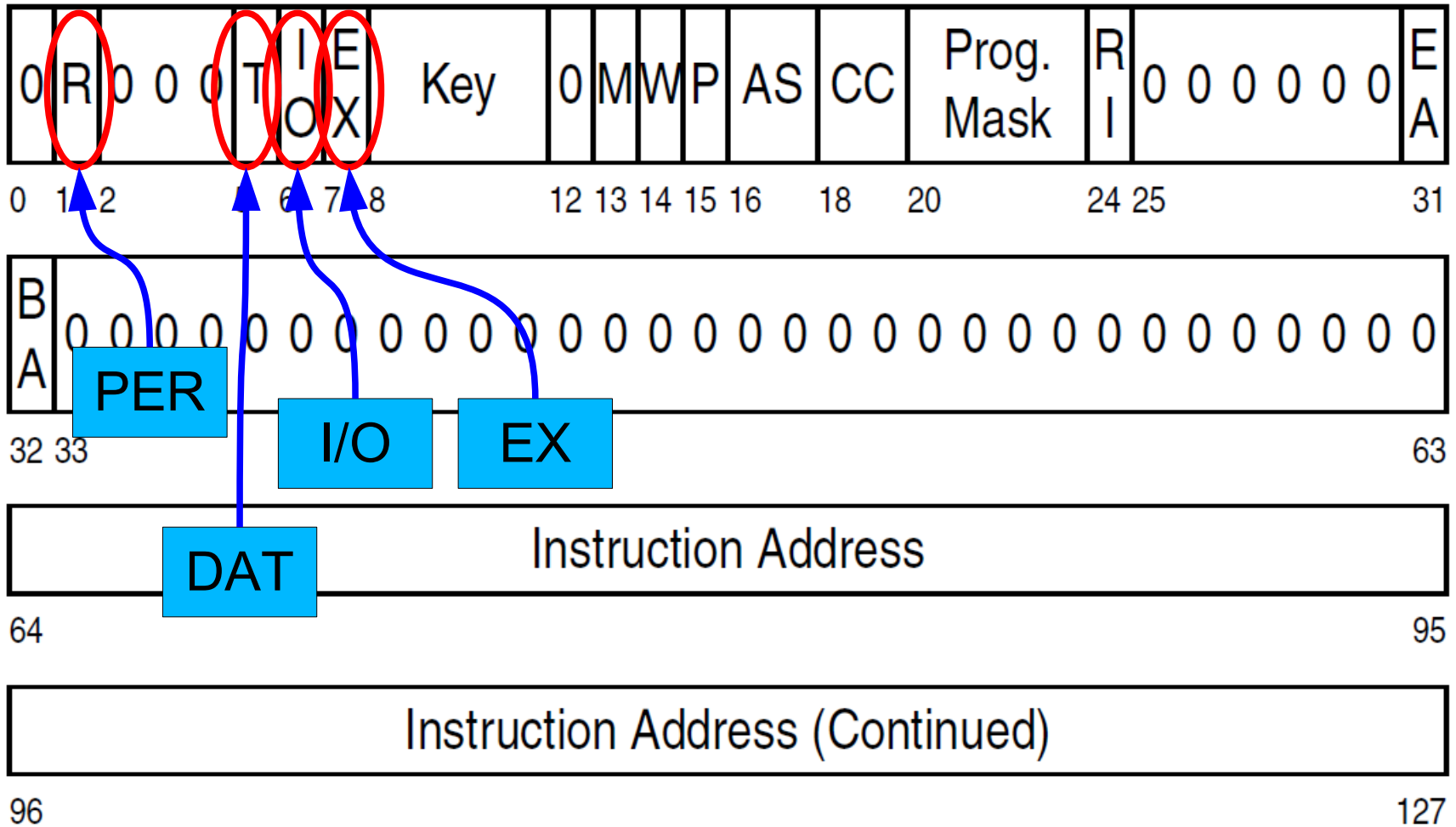


## Program Status Word (PSW) – Flag bits

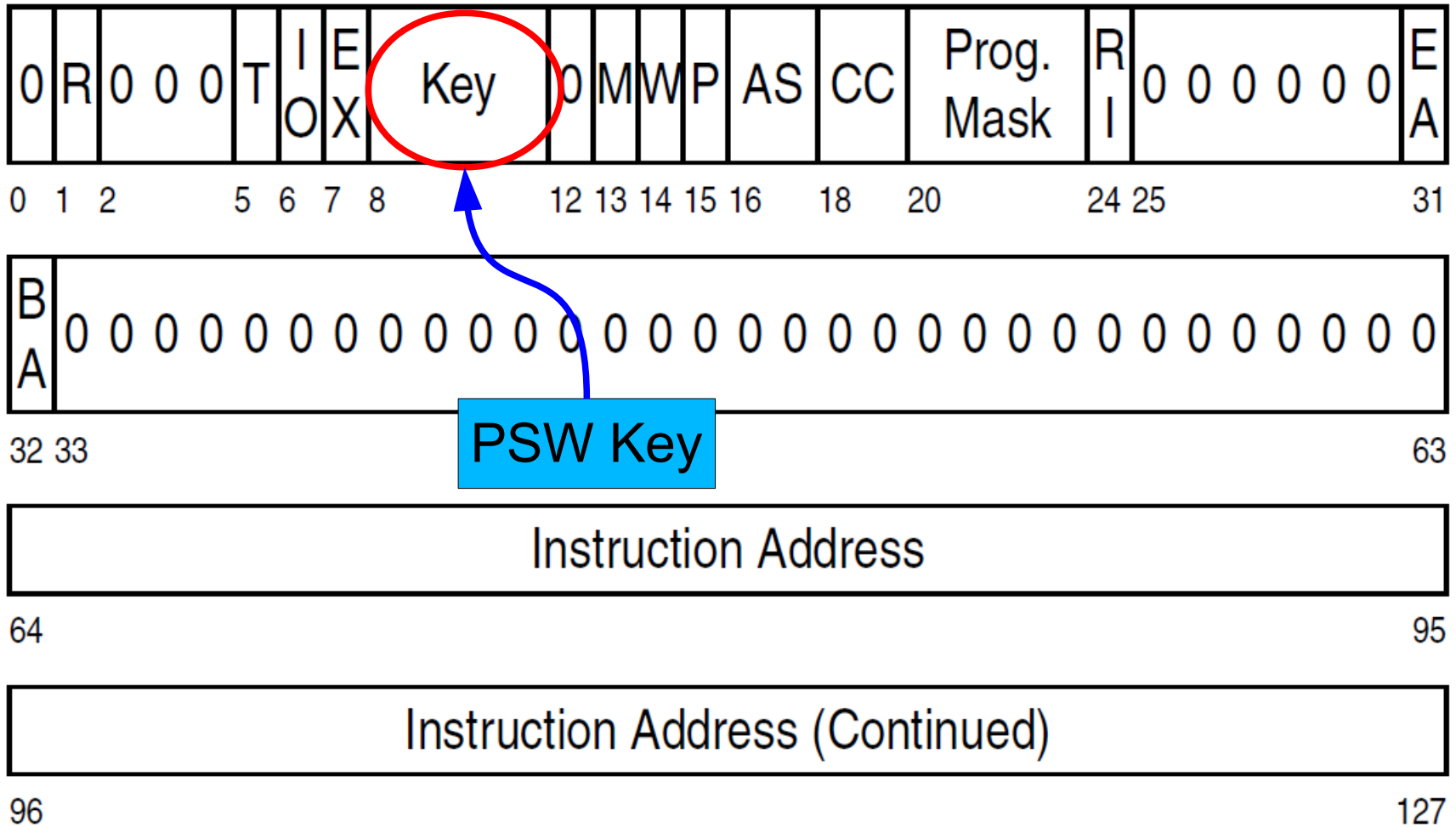
(A bit value of 1 indicates that the CPU is enabled for a function unless stated otherwise)

- Bit 1 – Program Event Recording (PER) Mask
  - Controls whether the CPU is enabled for interrupts associated with PER
- Bit 5 – DAT Mode
  - Controls whether the CPU has Dynamic Address Translation turned on
- Bit 6 – I/O Mask
  - Controls whether the CPU is enabled for interrupts
- Bit 7 – External Mask
  - Controls whether the CPU is enabled for external interrupts

## Program Status Word (PSW) – Flag bits



## Program Status Word (PSW) – PSW Key



## Program Status Word (PSW) – PSW Key

- Bits 8-11 are used to represent the PSW Key (value of 0-15)
- PSW Keys are used to provide a security mechanism over various regions of memory with key 0 being the most secure
- Whenever an instruction attempts to access a storage location that is protected against that type of reference (read/write of storage) and the storage key does not match the access key, a protection exception is recognised.
- Programs running in PSW key 0 have read write access to storage in every storage key
- Programs in keys 1 – 15 have read access to:
  - Storage which matches their PSW key
  - Storage (in any key) that's not fetch protected
  - Storage in key 9 if the hardware feature “subsystem storage protection override” is installed
- Programs in keys 1-15 have write access to:
  - Storage whose key matches their PSW key
  - Storage in key 9 if subsystem storage protection override is installed

## Program Status Word (PSW) – PSW Key – Manipulation

- IPK – Insert PSW Key
  - Used to *insert* the PSW Key *into* register 2
  - Used to store a copy of the current PSW Key typically before a switch to another key.
  - Bits 56-59 of register 2 are updated to contain the PSW Key, bits 60-63 are set to 0 and all other bits remain unmodified
- IPK cannot be used when bit 36 of CR0 is set to 0 and in problem state
- SPKA – Set PSW Key from Address
  - Used to set the PSW Key from an *address value*
  - Bits 56-59 of the 2<sup>nd</sup> operand are inserted into the PSW Key
- SPKA can only be used to set a key to which the current task is allowed to set a key determined by the PSW Key mask in CR3
- Both IPK and SPKA are privileged instructions

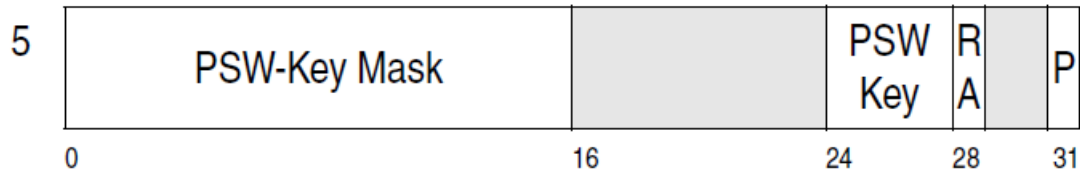
## Program Status Word (PSW) – PSW Key – Manipulation

- MVCK – Move with Key
  - Moves an operand with an access key specified as part of the instruction
  - If the program is not enabled to use that access key, then a privileged operation exception is raised
  - Can be a slow instruction
  
- BSA – Branch and Set Authority
  - Used to branch to another place in code and set the PSW key at the same time
  - Works as a flip-flop branching from “base authority” state to “reduced authority” state

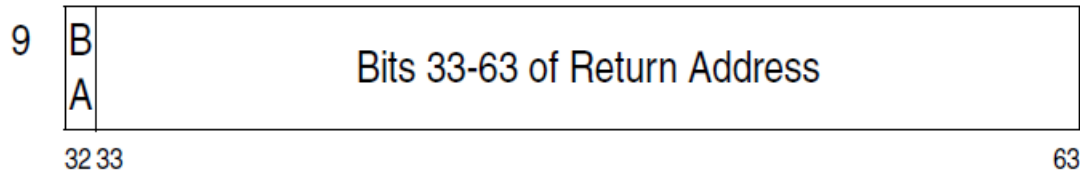
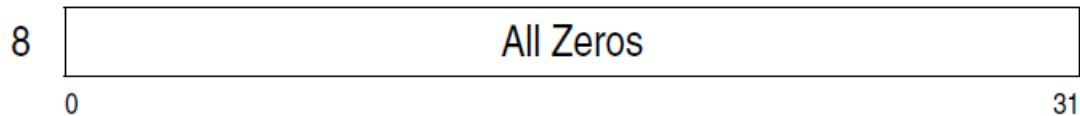
## Program Status Word (PSW) – PSW Key – BSA Operation

- BSA – Branch and Set Authority – example scenario
  - A service routine, e.g. a middleware service begins in the *base authority* state
  - The routine issues a BSA to switch to running a user routine
  - The user routine runs in *reduced authority* state
  - When the user routine wants to invoke the middleware service, it issues a BSA which branches back to a fixed location in the middleware and the state is returned to running in base authority state
- The control of the states is determined by the Dispatchable Unit Control Table (DUCT)

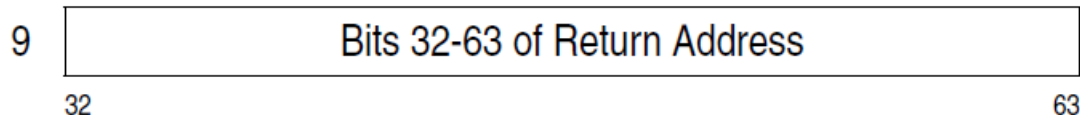
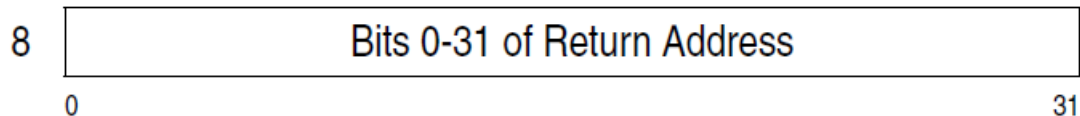
# Program Status Word (PSW) – PSW Key – BSA Operation - DUCT



## In the 24-Bit or 31-Bit Addressing Mode

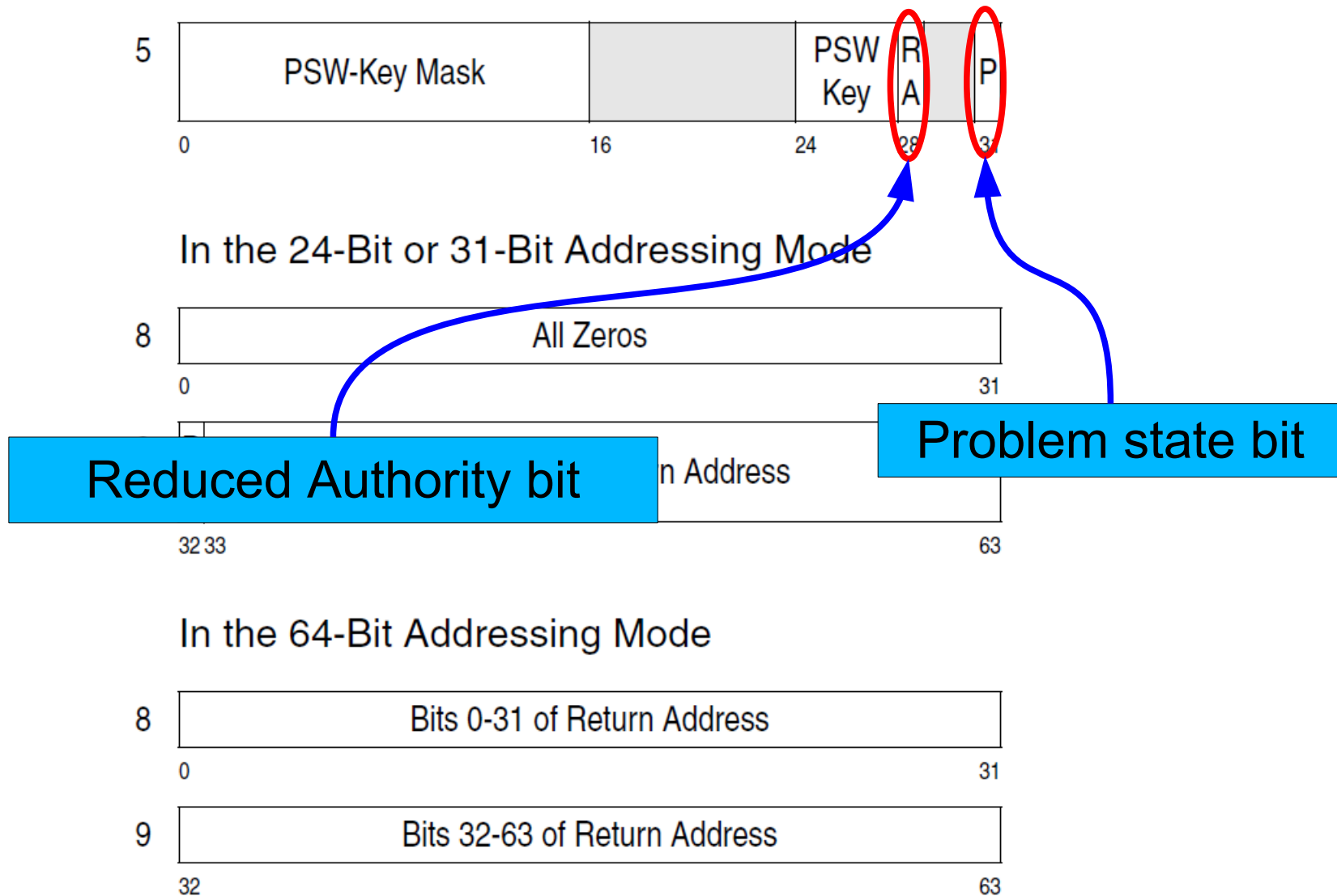


## In the 64-Bit Addressing Mode





# Program Status Word (PSW) – PSW Key – BSA Operation



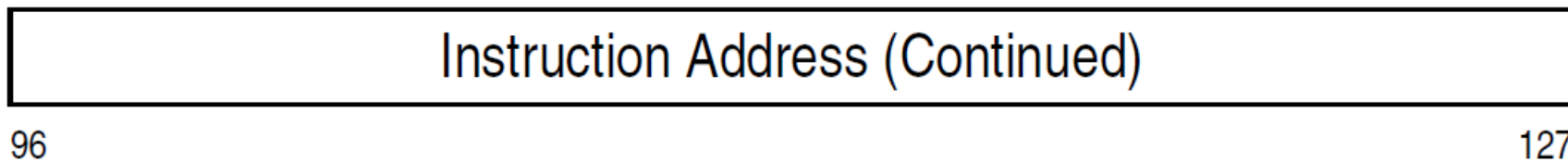
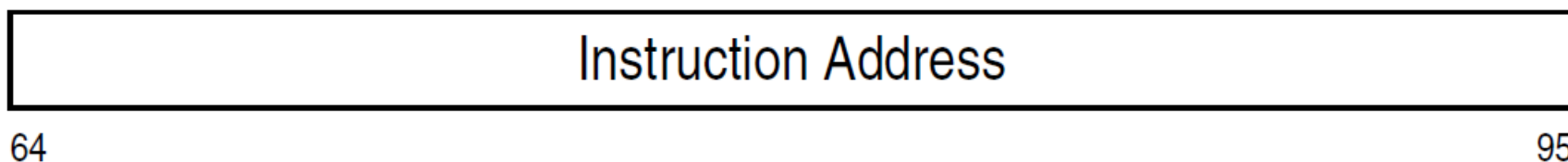
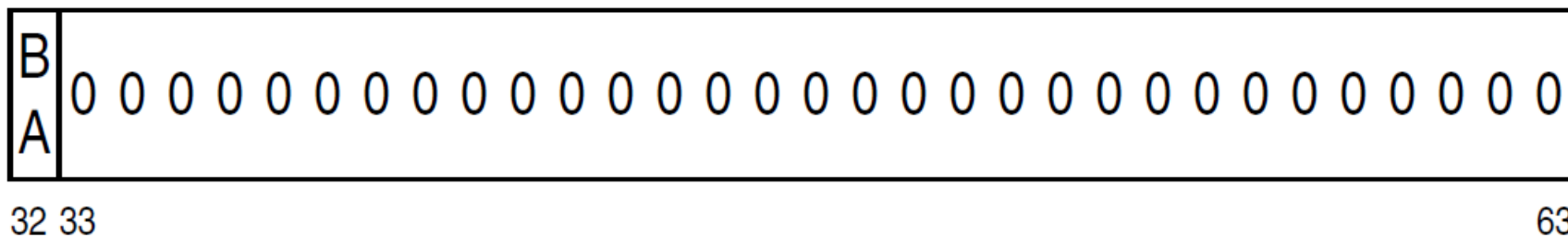
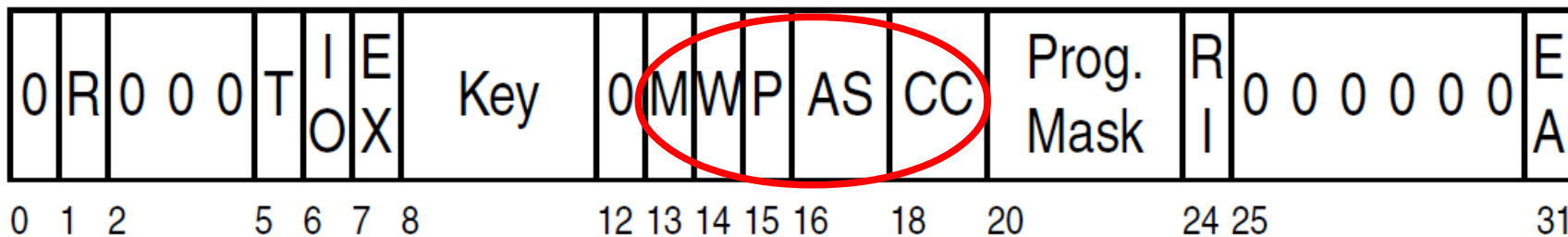
## Program Status Word (PSW) – PSW Key – BSA Operation – BA

- When the BSA instruction is used in base authority, the following is stored in the DUCT:
  - The PSW-key Mask (from CR3)
  - The current PSW Key
  - Problem state bit
  - The return address
  
- BSA then sets the Reduced Authority bit (RA) to 1 and loads:
  - The PSW-key Mask into CR3 from operand 1
  - The PSW Key from operand 1
  - The branch address into the PSW

## Program Status Word (PSW) – PSW Key – BSA Operation – RA

- When the BSA instruction is used in reduced authority, the following is restored from the DUCT:
  - The PSW-key Mask (to CR3)
  - The current PSW Key (to the PSW)
  - Problem state bit (to the PSW)
  - The return address (to the PSW – therefore the machine branches...)
  
- BSA then sets the Reduced Authority bit (RA) to 0

## Program Status Word (PSW) – More flag bits



## Program Status Word (PSW) – More flag bits

- Bit 13 – Machine Check Mask
  - Controls whether the CPU is enabled for interrupts by machine check conditions
- Bit 14 – Wait State
  - If on, the machine is waiting and no instructions are processed but interrupts may take place.
- Bit 15 – Problem State
  - The machine operates in two states – problem state (used for user code) and supervisor state (used for privileged code)
  - If an attempt is made to execute a privileged instruction in problem state, then a privileged operation exception occurs.  
Some instructions are *semi-privileged* and may or may not be permitted to execute in problem state depending on the outcome of other flags
  - All instructions are valid in supervisor state

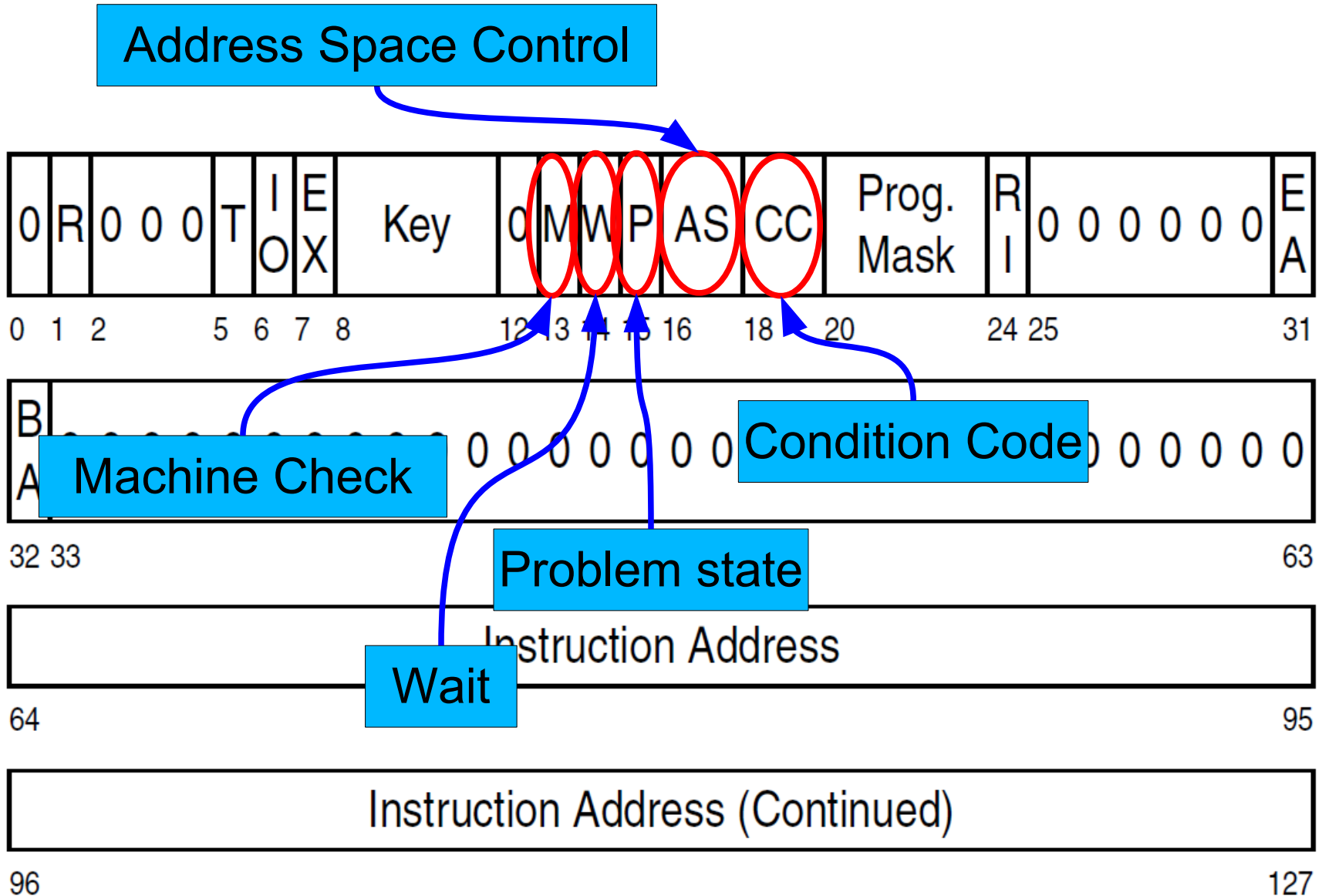
## Program Status Word (PSW) – More flag bits

- Bit 16-17 – Address-Space Control
  - Determines how addresses are handled in conjunction with bit 5 (DAT) via the following table:

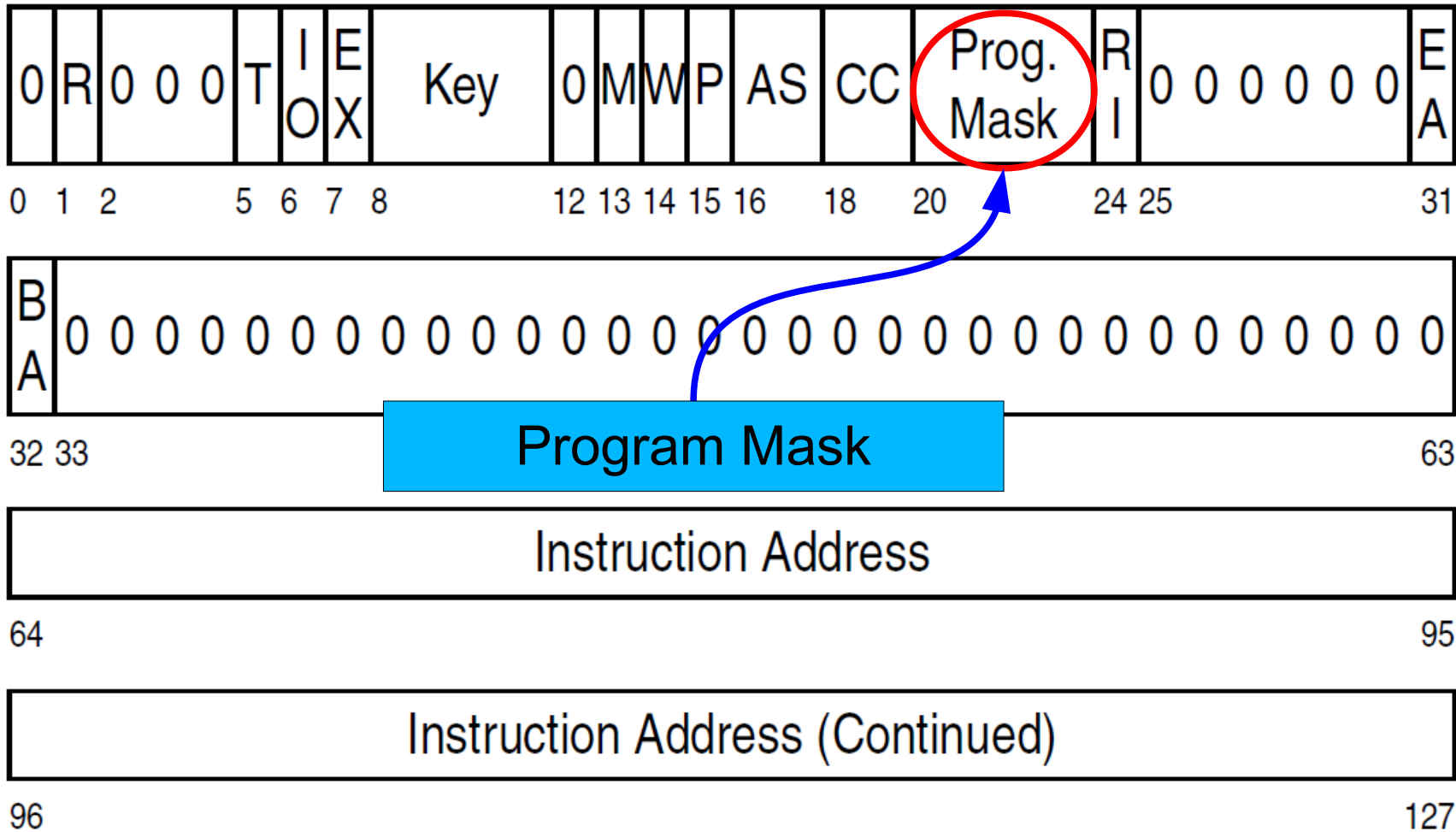
5	16	17	DAT	Mode	Instruction Addresses	Logical Addresses
0	0	0	Off	Real Mode	Real	Real
0	0	1	Off	Real Mode	Real	Real
0	1	0	Off	Real Mode	Real	Real
0	1	1	Off	Real Mode	Real	Real
1	0	0	On	Primary-space Mode	Primary virtual	Primary virtual
1	0	1	On	Access-register Mode	Primary virtual	AR specified vrt
1	1	0	On	Secondary-space Mode	Primary virtual	Secondary vrt
1	1	1	On	Home-space Mode	Home virtual	Home virtual

- Bits 18-19 – Condition Code

## Program Status Word (PSW) – More flag bits



# Program Status Word (PSW) – PSW Key – Program Mask





## Program Status Word (PSW) – Program Mask

- Bits 20-23 – Program Mask
  - Controls a set of program exceptions
  - When the corresponding bit is on, the exception results in an interrupt

Program Mask PSW bit	Program Exception
20	Fixed-point overflow
21	Decimal overflow
22	HFP exponent underflow
23	HFP significance

- The Program Mask can be manipulated by using the instruction SET PROGRAM MASK (SPM)
- The contents of the Program Mask can be examined using the instruction INSERT PROGRAM MASK (IPM)

## Program Status Word (PSW) – Using the PSW for debugging

- The PSW stores invaluable information about the general state of the machine during a program's execution
- The most interesting time to examine a PSW is when something goes wrong. Even a summary dump will provide the programmer with:
  - The contents of the PSW
  - The contents of the general purpose registers
  - The next instruction to be executed

## Program Status Word (PSW) – Something has gone wrong

```
SYSTEM COMPLETION CODE=0C4 REASON CODE=00000010
TIME=10.58.19 SEQ=01263 CPU=0000 ASID=002C
PSW AT TIME OF ERROR 078D0000 80007FF6 ILC 4 INTC 10
ACTIVE LOAD MODULE ADDRESS=00007FF0 OFFSET=00000006
NAME=GO
DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C00607FE
GR 0: FD000008 1: 00006FF8
2: 00000040 3: 008DAD6C
4: 008DAD48 5: 008FF130
6: 008CAFC8 7: FD000000
8: 008FCE28 9: 008D88F0
A: 00000000 B: 008FF130
C: FD000008 D: 00006F60
E: 80FDA688 F: 80007FF0
END OF SYMPTOM DUMP
```

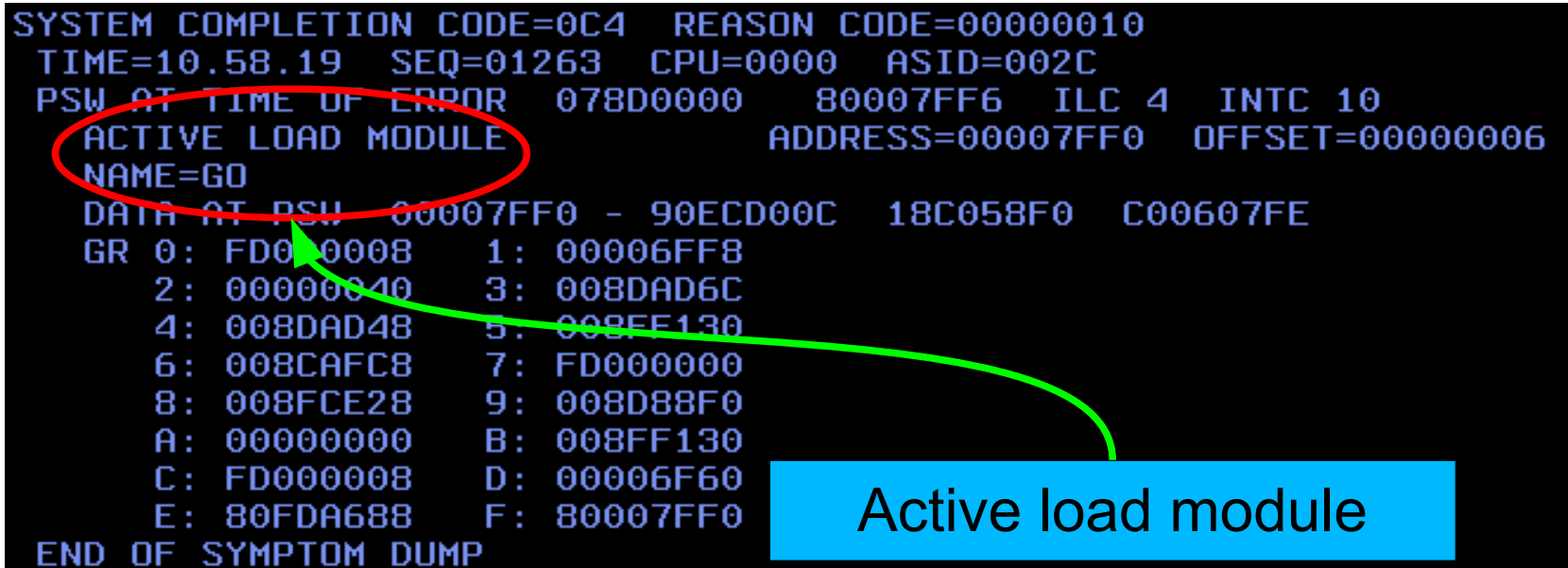
- Running a job has resulted in an 0C4 ABEND occurring. The summary dump in the job may be enough information to work out what has gone wrong.

## Program Status Word (PSW) – Something has gone wrong

```

SYSTEM COMPLETION CODE=0C4 REASON CODE=00000010
TIME=10.58.19 SEQ=01263 CPU=0000 ASID=002C
PSW AT TIME OF ERROR 078D0000 80007FF6 ILC 4 INTC 10
ACTIVE LOAD MODULE ADDRESS=00007FF0 OFFSET=00000006
NAME=GO
DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C00607FE
GR 0: FD000008 1: 00006FF8
2: 00000040 3: 00BDAD6C
4: 00BDAD48 5: 008FE130
6: 00BCAFC8 7: FD000000
8: 00BFCE28 9: 008D88F0
A: 00000000 B: 008FF130
C: FD000008 D: 00006F60
E: 80FDA688 F: 80007FF0
END OF SYMPTOM DUMP

```



- First, look at the active load module
  - In this example, the load module name is GO since the LKEDG JCL procedure was used. From this we already know that the error occurred in our load module and not in either the assembler, linkage editor nor other part of z/OS

## Program Status Word (PSW) – Something has gone wrong

```

SYSTEM COMPLETION CODE=0C4 REASON CODE=00000010
TIME=10.58.19 SEQ=01263 CPU=0000 RSTD=002C
PSW AT TIME OF ERROR 078D0000 80007FF6 ILC 4 INTC 10
ACTIVE LOAD MODULE ADDRESS=00007FF0 OFFSET=00000006
NAME=GO
DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C0607FE
GR 0: FD000008 1: 00006FF8
B: 008CFE28 7: FD000000
8: 008FCE28 9: 008D
A: 00000000 B: 008F
C: FD000008 D: 00006FF8
E: 80FDA688 F: 80007FF0
END OF SYMPTOM DUMP

```

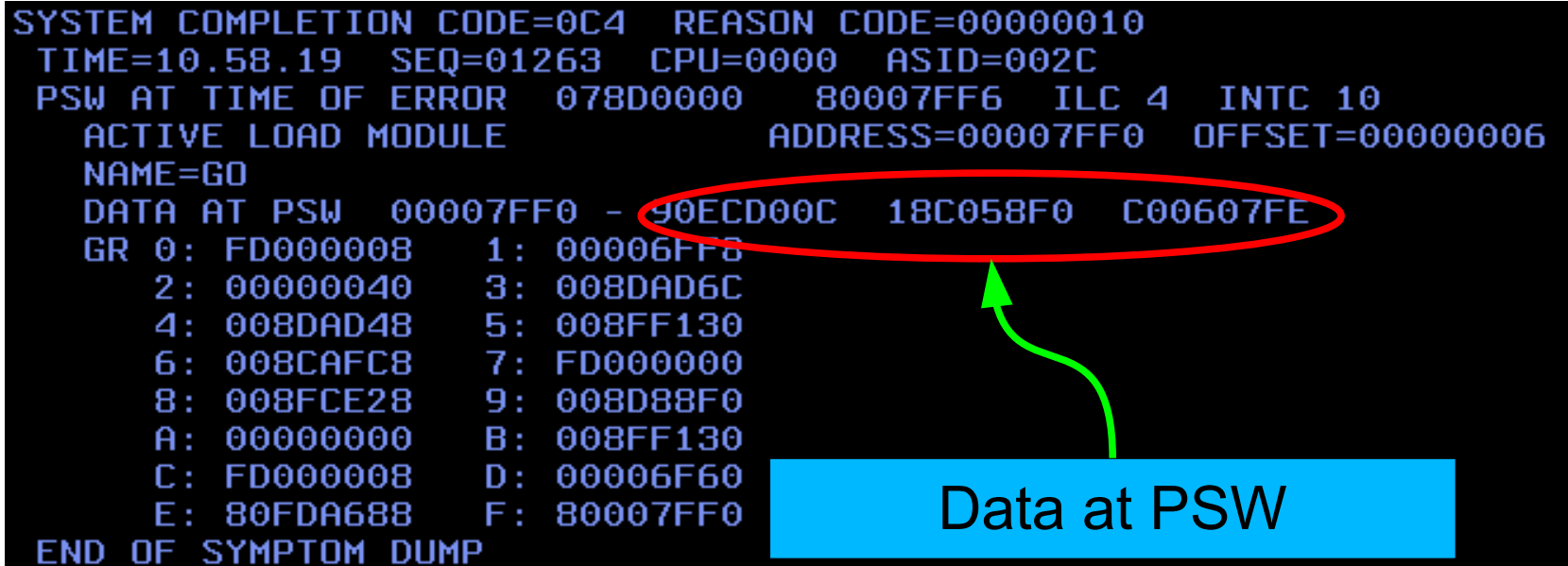
Address in PSW

Address of load module and offset

- Next, double check the information in the PSW against the other information in the summary dump
  - The PSW shows that the next instruction address to be executed is X'7FF6'
  - This agrees with the data in the dump showing the address of the load module (X'7FF0') and the offset into the load module (X'0006')

## Program Status Word (PSW) – Something has gone wrong

```
SYSTEM COMPLETION CODE=0C4 REASON CODE=00000010
TIME=10.58.19 SEQ=01263 CPU=0000 ASID=002C
PSW AT TIME OF ERROR 078D0000 80007FF6 ILC 4 INTC 10
ACTIVE LOAD MODULE ADDRESS=00007FF0 OFFSET=00000006
NAME=GO
DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C00607FE
GR 0: FD000008 1: 00006FF8
2: 00000040 3: 008DAD6C
4: 008DAD48 5: 008FF130
6: 008CAFC8 7: FD000000
8: 008FCE28 9: 008D88F0
A: 00000000 B: 008FF130
C: FD000008 D: 00006F60
E: 80FDA688 F: 80007FF0
END OF SYMPTOM DUMP
```



The image shows a system dump with the following text:

```
SYSTEM COMPLETION CODE=0C4 REASON CODE=00000010
TIME=10.58.19 SEQ=01263 CPU=0000 ASID=002C
PSW AT TIME OF ERROR 078D0000 80007FF6 ILC 4 INTC 10
ACTIVE LOAD MODULE ADDRESS=00007FF0 OFFSET=00000006
NAME=GO
DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C00607FE
GR 0: FD000008 1: 00006FF8
2: 00000040 3: 008DAD6C
4: 008DAD48 5: 008FF130
6: 008CAFC8 7: FD000000
8: 008FCE28 9: 008D88F0
A: 00000000 B: 008FF130
C: FD000008 D: 00006F60
E: 80FDA688 F: 80007FF0
END OF SYMPTOM DUMP
```

The data at the PSW is highlighted with a red oval, and a green arrow points from a blue box labeled "Data at PSW" to the highlighted data.

- The data at the PSW shows the instructions which were, are being, and will be executed

## Program Status Word (PSW) – Something has gone wrong

**DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C00607FE**

```

000000      00000 00010      15 *
                                16 * MAIN PROGRAM STARTS HERE
                                17 *
000000      90EC D00C      0000C      18 EX1      CSECT
000004      18C0                                19 EX1      AMODE 31
                                20 EX1      RMODE 24
                                21 * USUAL PROGRAM SETUP
                                22          STM   R14,R12,12(R13)
                                23          LR    R12,0
                                24          USING *,12
000006      58F0 C006      0000C      25 * SET RETURN CODE IN REGISTER 15
                                26          L     R15,RET_CODE
00000A      07FE                                27 * RETURN
                                28          BR   R14
                                29 * *****
                                30 * END OF PROGRAM
                                31 * *****
00000C      00000000      32 RET_CODE  DC   F'0'
000010                                33          LTORG ,

```

- Examining the program listing at offset 6 shows where the error occurred. Using the data at the PSW and looking at the machine code generated by HLASM in the listing confirms this and that so far our diagnosis of the problem is correct

## Program Status Word (PSW) – Something has gone wrong

DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C00607FE

Offset 6

- Examining the program listing at offset 6 shows where the error occurred. Using the data at the PSW and looking at the machine code generated by HLASM in the listing confirms this and that so far our diagnosis of the problem is correct
- We now know the instruction which caused the error was:  
 58F0 C006 → L R15,RET\_CODE



## Program Status Word (PSW) – Something has gone wrong

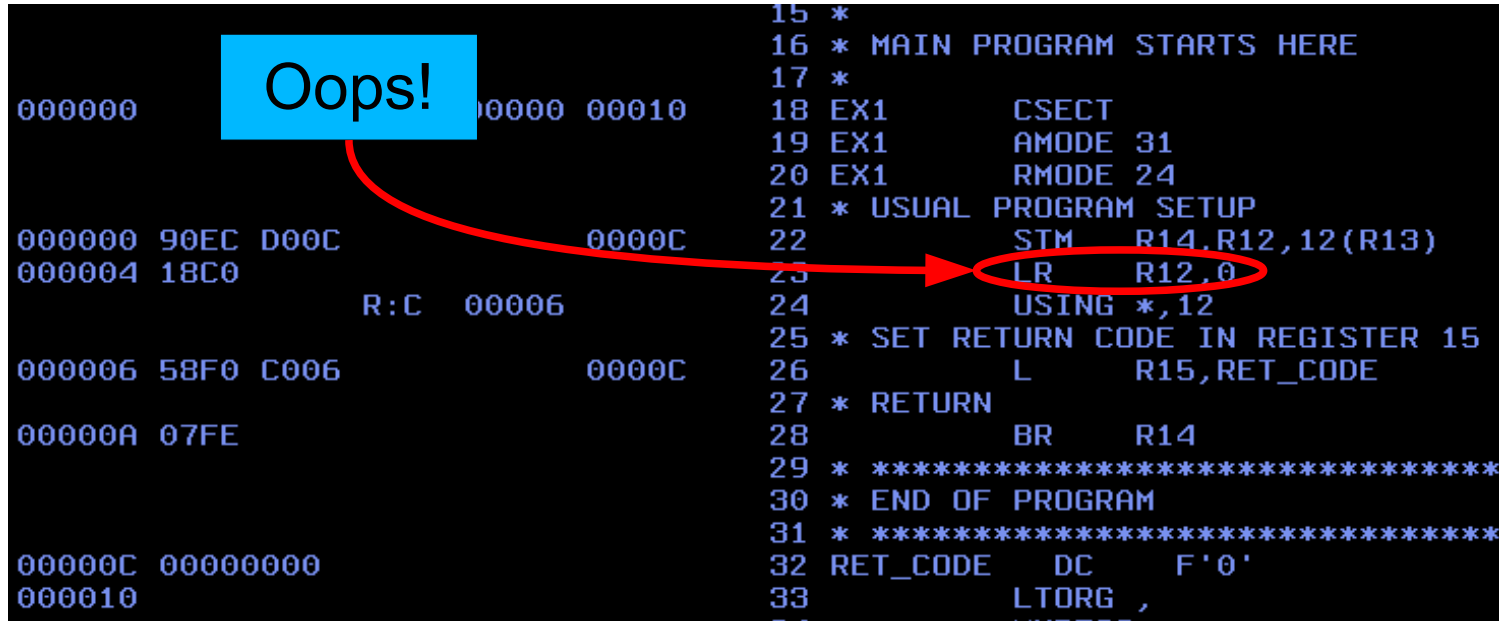
- At this stage of our debugging we know:
  - The load module name that caused the error
  - The offset into the load module at which the error occurred
- We have also double-checked that what was printed in the summary dump is confirmed by the data at the PSW
- Examining the instruction at fault, we determine the following:
  - 58F0 C006 → L R15,RET\_CODE
  - 58 – OPICODE = LOAD
  - F – Register 15, the register to be loaded
  - 0 – Index register (unused since it has a value of 0)
  - C – Base register is register 12
  - 006 – Displacement from the base register from which the data will be loaded
- So, the instruction is attempting to load register 15 with the contents of memory at an offset of 6 from register 12.

## Program Status Word (PSW) – Something has gone wrong

```
SYSTEM COMPLETION CODE=0C4 REASON CODE=00000010
TIME=10.58.19 SEQ=01263 CPU=0000 ASID=002C
PSW AT TIME OF ERROR 078D0000 80007FF6 ILC 4 INTC 10
ACTIVE LOAD MODULE ADDRESS=00007FF0 OFFSET=00000006
NAME=GO
DATA AT PSW 00007FF0 - 90ECD00C 18C058F0 C00607FE
GR 0: FD000008 1: 00006FF8
 2: 00000040 3: 008DAD6C
 4: 008DAD48 5: 008FF130
 6: 008CAFC8 7: FD000000
 8: 008FCE28 9: 008D88F0
 A: 00000000 B: 008FF130
 C: FD000008 D: 00006F60
 E: 80FDA688 F: 80007FF0
END OF SYMPTOM DUMP
```

- The summary dump also shows the contents of the general purpose registers
- The value in register 12 is X'FD000008'
- The instruction at fault is attempting to load a value from address X'FD00000E' – which is unaddressable by our program and therefore the cause of the error
- Note that the value of register 12 is the same as the value of register 0...

## Program Status Word (PSW) – Something has gone wrong



```

000000 000000 00010 15 *
                                16 * MAIN PROGRAM STARTS HERE
                                17 *
000000 90EC D00C 0000C 18 EX1 CSECT
000004 18C0 00006 19 EX1 AMODE 31
                                20 EX1 RMODE 24
                                21 * USUAL PROGRAM SETUP
                                22 STM R14,R12,12(R13)
                                23 LR R12,0
                                24 USING *,12
000006 58F0 C006 0000C 25 * SET RETURN CODE IN REGISTER 15
                                26 L R15,RET_CODE
00000A 07FE 0000C 27 * RETURN
                                28 BR R14
                                29 * *****
                                30 * END OF PROGRAM
                                31 * *****
00000C 00000000 0000C 32 RET_CODE DC F'0'
000010 00000000 0000C 33 LTORG ,

```

- Looking back through the program, we can see that register 12 was loaded with the value of register 0 during program startup
- It looks as if the programmer made a typo and instead of using LR 12,0 should have used BALR 12,0 in order to load the address of the next instruction into register 12. This would make sense since they are using register 12 to establish addressability to the program's data
- Correcting this mistake fixes the program

## Program Status Word (PSW) – Using the PSW for debugging

- Using the data in the summary dump, the PSW and the program listing has allowed us to fix the cause of the error.
- Although this is a simple example, it demonstrates the basics of debugging assembler problems.