

HLASM

IBM's High Level Assembler

Calling Conventions

Static Linkage – Part I

24-bit and 31-bit addressing modes

LANGUAGE

High Level Assembler for z/OS & z/VM & z/VSE

Version 1 Release 6

REFERENCES

- *HLASM Programmer's Guide*, SC26-4941-08, 2017
- *IBM z/Architecture Principles of Operation*, SA22-7832-12, 2019
- *MVS Programming: Assembler Services Guide*, SA23-1368-40, 2019
- *Assembler Language Programming for IBM System z™ Servers*, 2nd edition 2016, by John Ehrman
- *Basic IBM Mainframe Assembly Language Programming, 2016* by Kevin C. O Kane

Question

How do we preserve states when control is passed from one program to another?

Goal

In this presentation we discuss status preservation.

We will discuss how to pass control to a subroutine, how we return to a calling program, and argument passing in Static Linkage Part II.

Scope

1. Primary mode programs
2. Code written with static save areas and linkage.
3. Programs that have static areas defined internally.
i.e. “traditional” or “standard” linkage

Note: Rentrant programs / recursion are out of scope

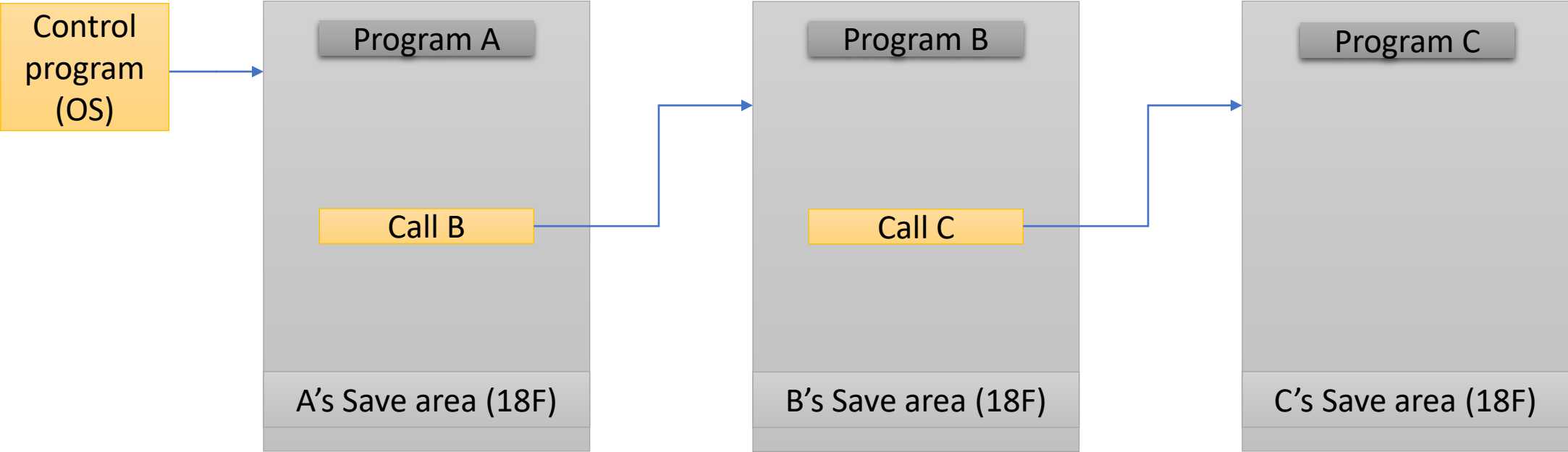
Primary mode programs

- Execute all of their instructions in primary ASC mode
- ASC = address space control
- Address Space (AS) = the range of addresses available to a computer program
- Examples of address spaces: a batch job, a TSO session, a CICS region
- ASC Mode = program uses the contents of GPRs to resolve an address to a specific location. The ASC mode of the called program determines where a program saves registers (calling program save area or the linkage stack).

Primary mode programs (cont.)

- ASC mode and access registers are important when using cross-memory services and communicating with another address space.
- Need APF and supervisor state.
- Application programmers will have access to neither.

Program A calls Program B, and Program B calls C



The Calling Process - Roles

The calling program (aka Caller)

- is a program that is calling another subroutine/program*
- it must know where to transfer control

The called program (aka Target or Callee)

- is the called subroutine/program
- it must know where to return control

* the words “subroutine” and “program” are used interchangeably

The Calling Process - Linkage

A set of conventions used by an operating system where programs:

1. call one another
2. pass arguments
3. return values

Key Items to Consider

1. Control how to pass control to a subroutine and return
2. Argument passing how to provide data needed by the subroutine and access its results
3. Status preservation how to ensure that nothing important is lost, modified, or destroyed in the process

Linkage - Status Preservation

The issues

1. What data/info should be preserved?
2. Who should do the preserving: the calling program or the target program?

“A primary mode program is one that executes all its instructions in primary ASC (address space control) mode and does not change the contents of ARs (access registers) 2 through 13.”

[z/OS MVS Assembler Services Guide, Chapter 2 - Linkage Conventions](#)

z/OS MVS Assembler Services Guide, Chapter 2 – Linkage Conventions

A calling program provides its target program with a 72-byte register save area unless the target program's interface requirements are otherwise specified. It is the caller's responsibility to provide a save area that meets the specifications provided by the target program.

The calling program obtains storage for the save area from its primary address space. The save area must begin on a word boundary.

Before invoking the target program, the calling program loads the address of the save area into general-purpose register 13.

etc.

John Ehrman, Section 37.4

[24- and 31-bit addressing modes]

By convention, the caller provides a “standard” 18-word save area, and its address is passed to the callee in GR13. The caller's general registers are stored starting at offset +12 in the order GR14, GR15, GR0, GR1, GR2, ..., GR12.

The easiest way to save the registers is to execute the instruction

```
STM 14,12,12(13)
```

This saves GR14-GR12 in caller's save area before the called program modifies any of them.

This [STM] is often one of the first instructions executed by a called program.

Linkage - 24-bit or 31-bit addressing mode

Every program that calls another has a local save area.

Lowest level programs (which don't call another) don't need a save area.

The target must save and restore the caller's register.

Concept the target can take advantage of its (possibly) economical use of registers by saving and restoring only the ones it modifies.

Linkage – High level process

Caller (A) ---> Target (B)

Program A calls B

Program B saves A's registers in A's save area (register preservation)

Program B stores A's save area address in its own save area (B to A chain)

Program B stores its save area address in A's save area (A to B chain)

Program B sets a base register and performs its work (local addressability)

Program B restores A's registers before returning control to A

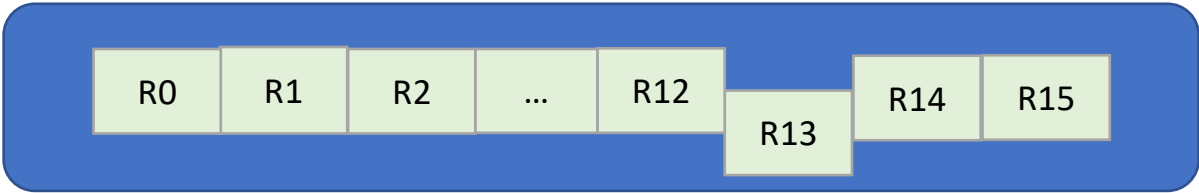
Note: this is a static save area model, meaning there is defined storage in each program for purposes of linkage (i.e. storage space is part of the programs; it is not dynamically allocated at run time)

Linkage – Registers convention

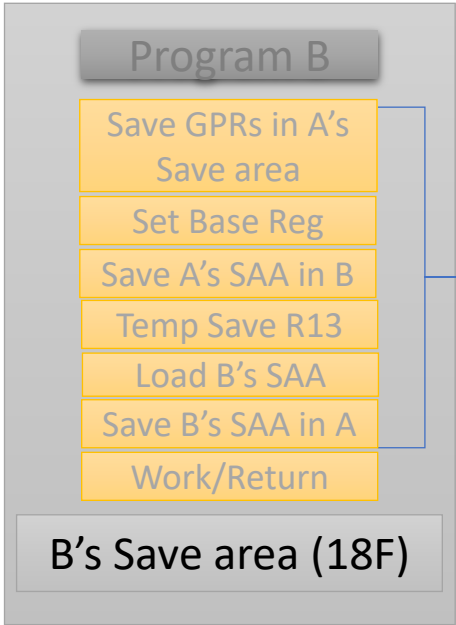
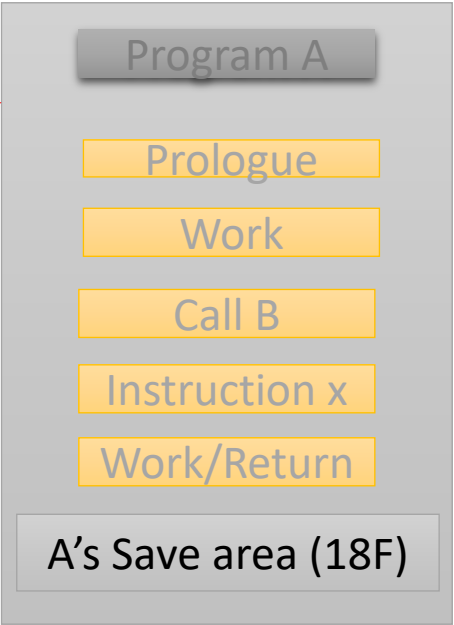
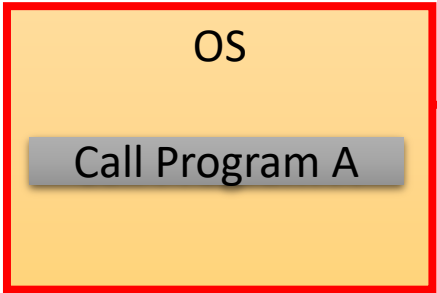
R0,R1	parameter registers, used by the CALL macro (as example) to pass parms (addresses of data) to the called program; a table of addresses in memory; each address points to a parameter; R1 holds the addresses; R1=0 if no parms are being passed
R12	used as typical base register for symbolic addressing
R13	save area register - address of caller's save area; called program stores caller's registers here; save area is 18 full words
R14	return register - address in caller's space; when finished, the called program branches here
R15	entry point register – address of the called program's entry point; the address of the first instruction in called program

OS Calls Program A

General Purpose
Registers (GPRs)

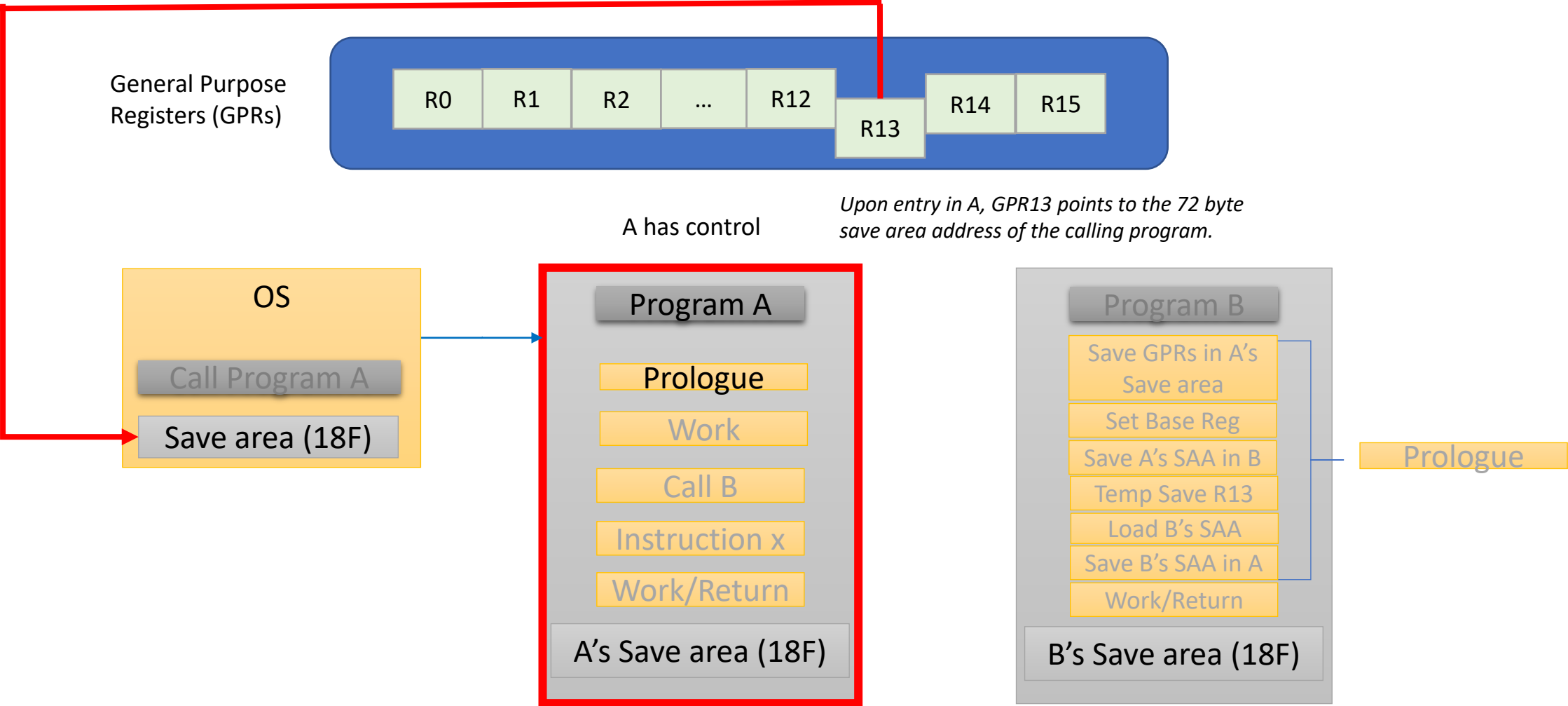


OS has control

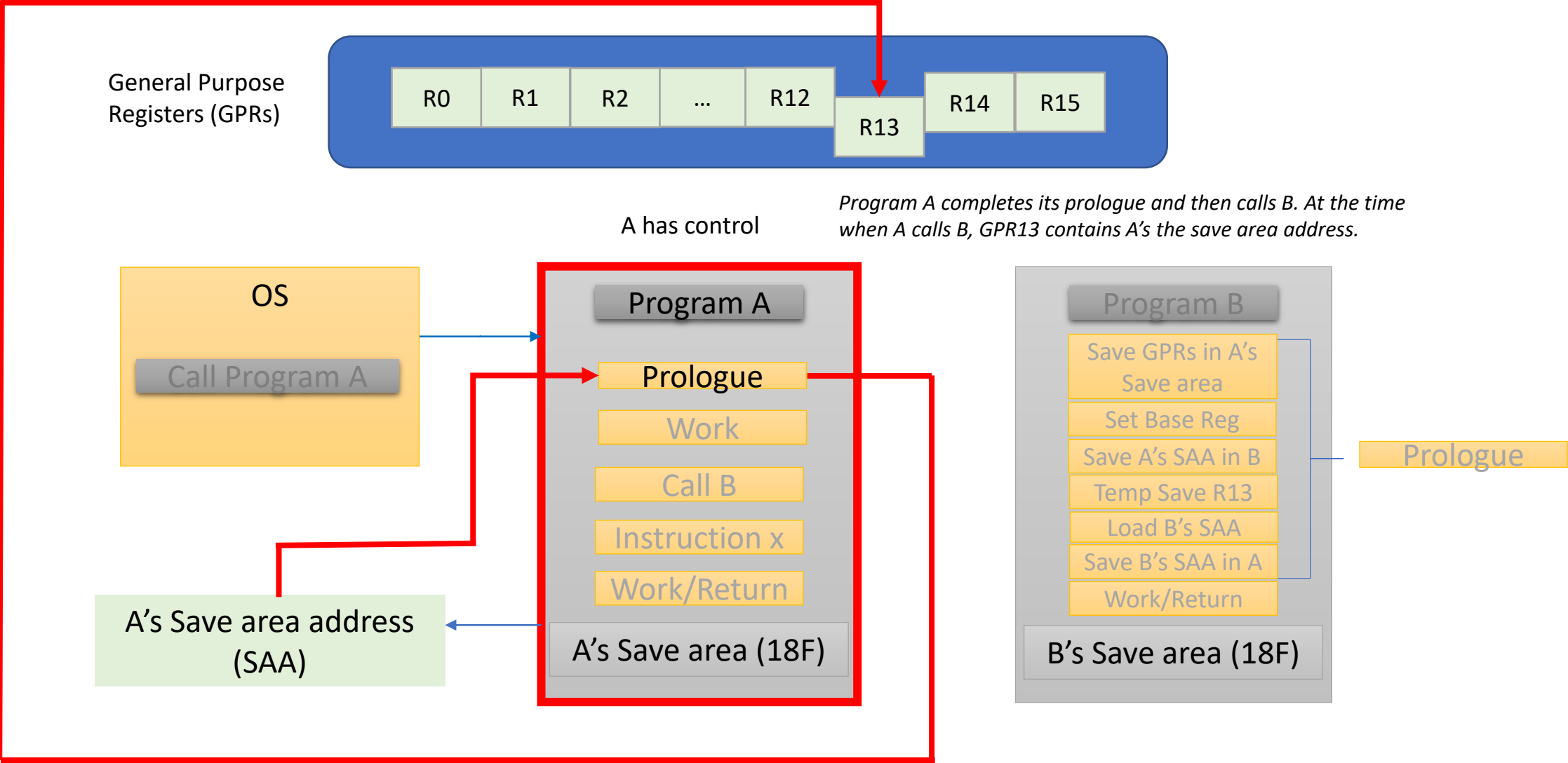


Prologue

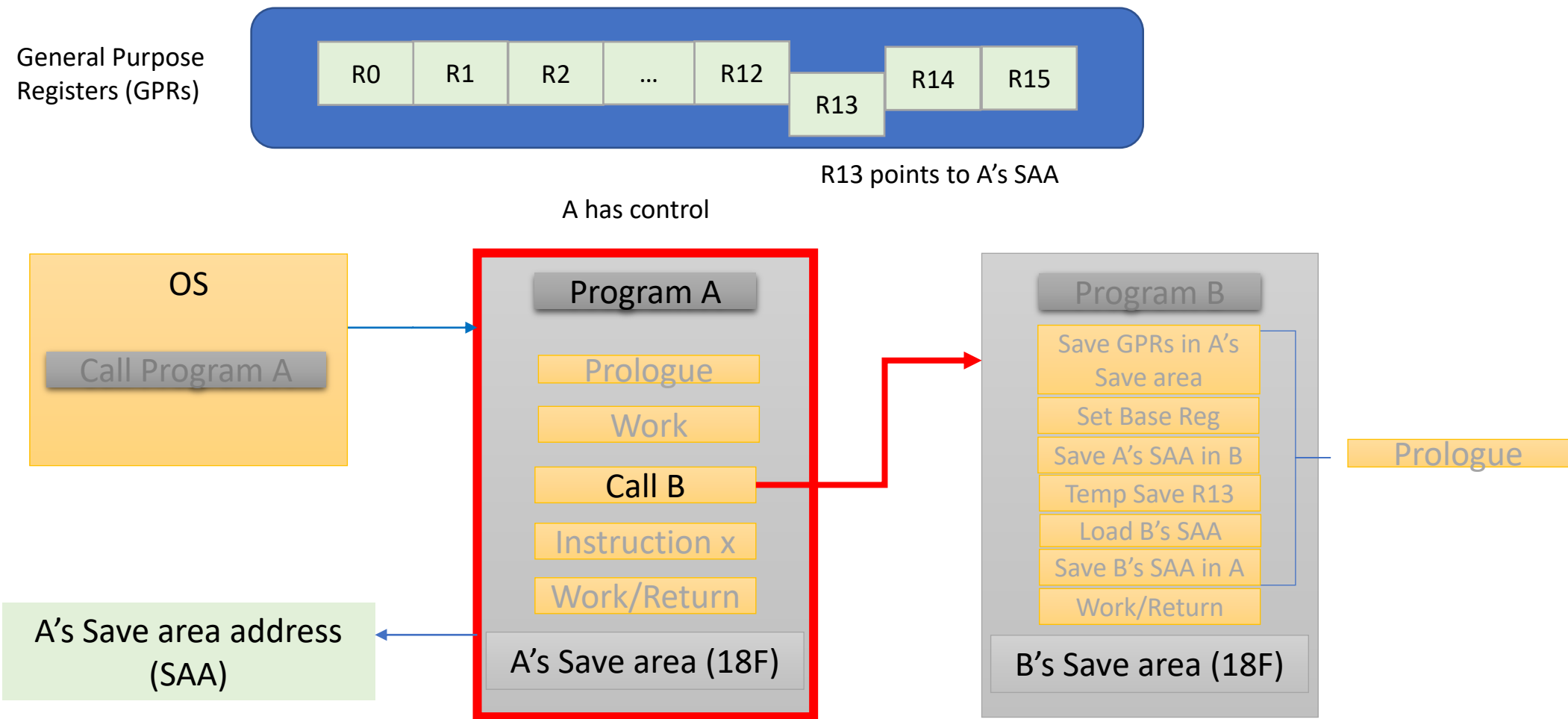
Program A completes its Prologue



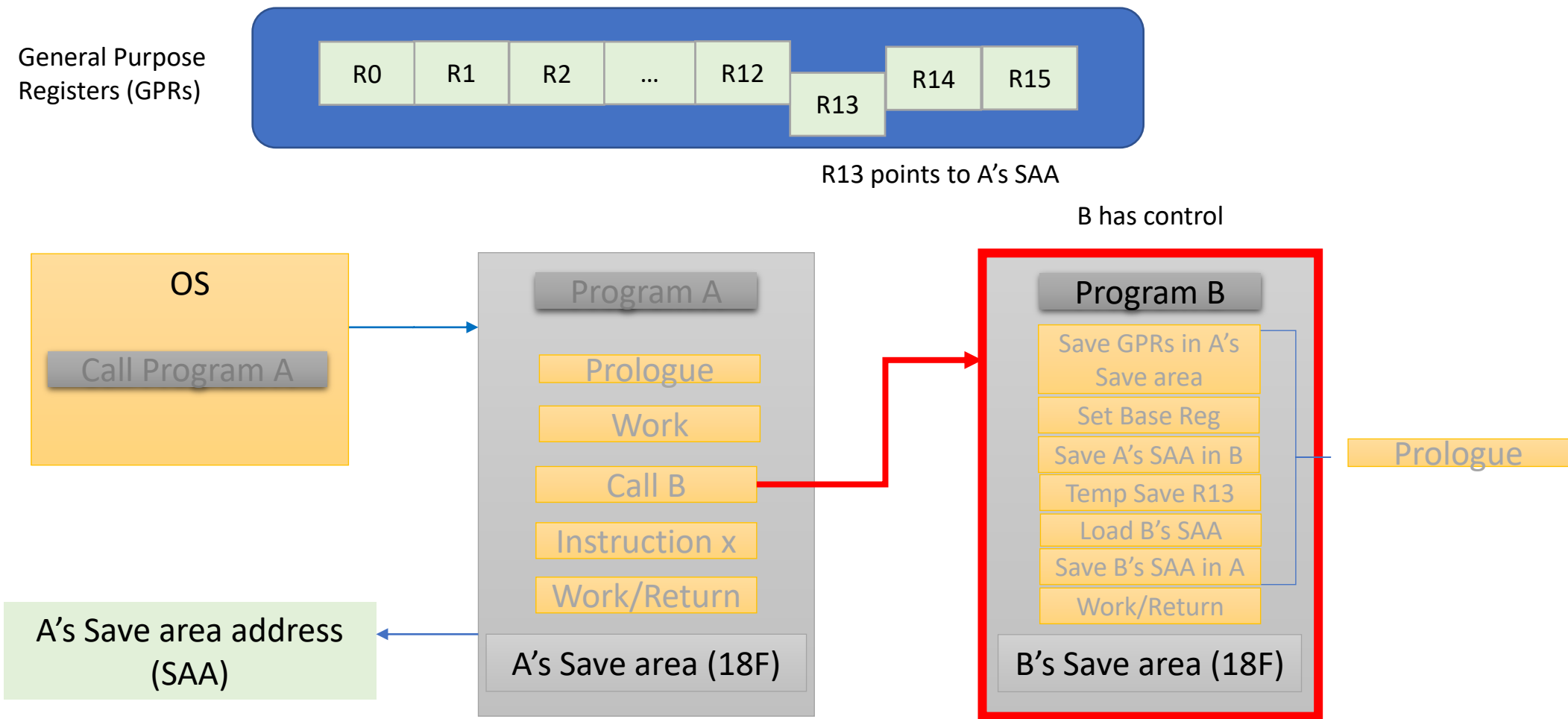
Program A completes its Prologue

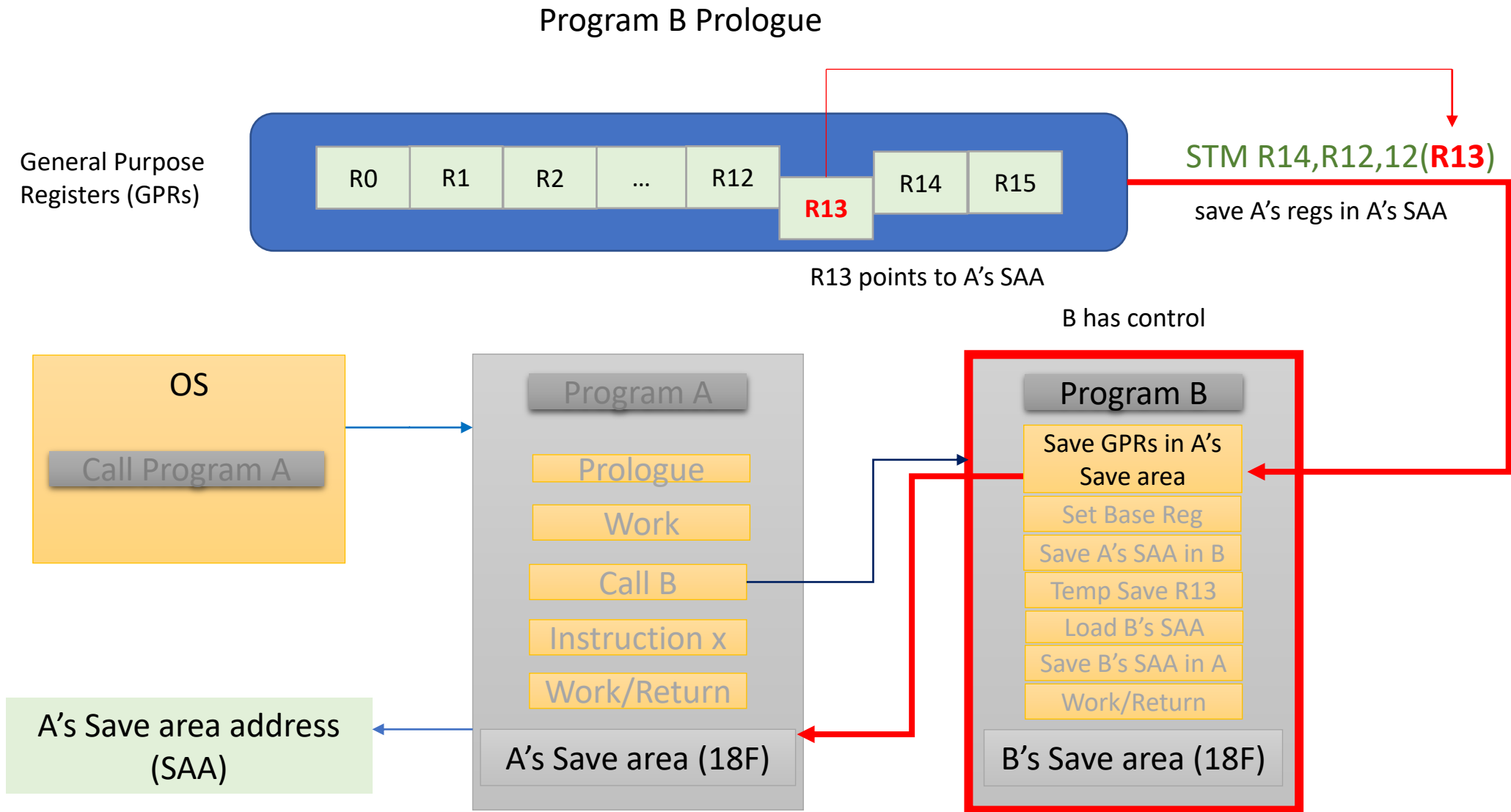


Program A calls Program B



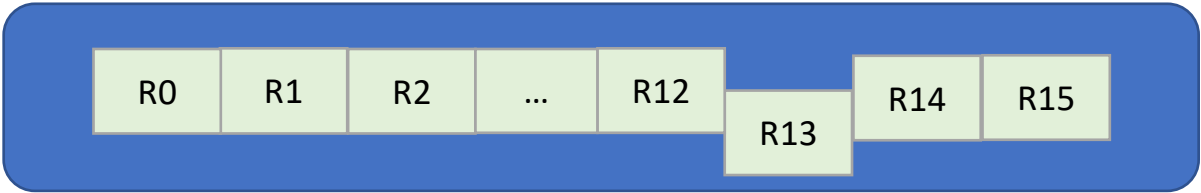
Program B Prologue





Program B Prologue

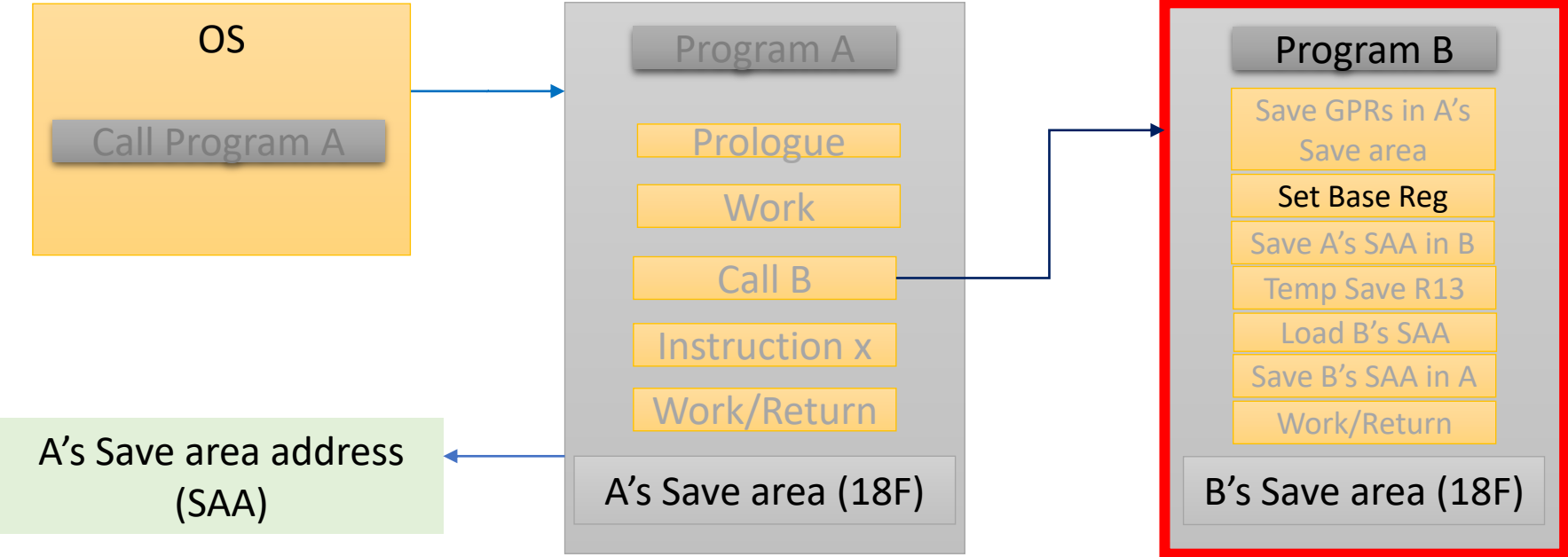
General Purpose
Registers (GPRs)



BASR R12,R0 **[*]**
USING *,R12
establish a base register in B

R13 points to A's SAA

B has control



[*] see **z Architecture Principles of Operation**, Appendix A for Linkage Instructions (BAL, BALR, BAS, BASR, BASSM, BSM)

Symbolic addressing

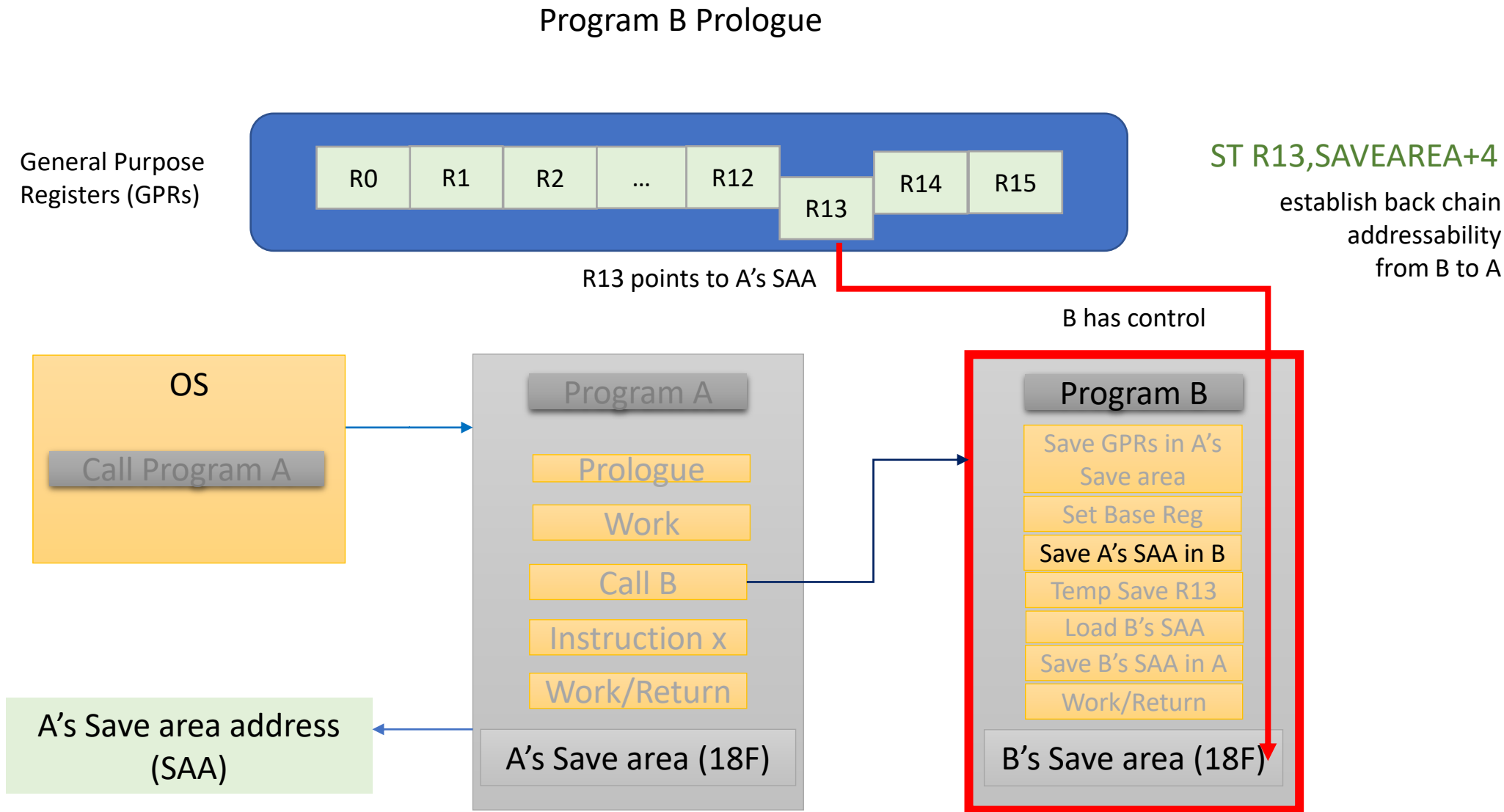
- Instruction operands and addresses can be accessed explicitly
D(B) where **D** is displacement from **B** (base register)
- However, we can let the assembler calculate displacements from a base register; use labels to refer to addresses symbolically (much simpler and efficient)

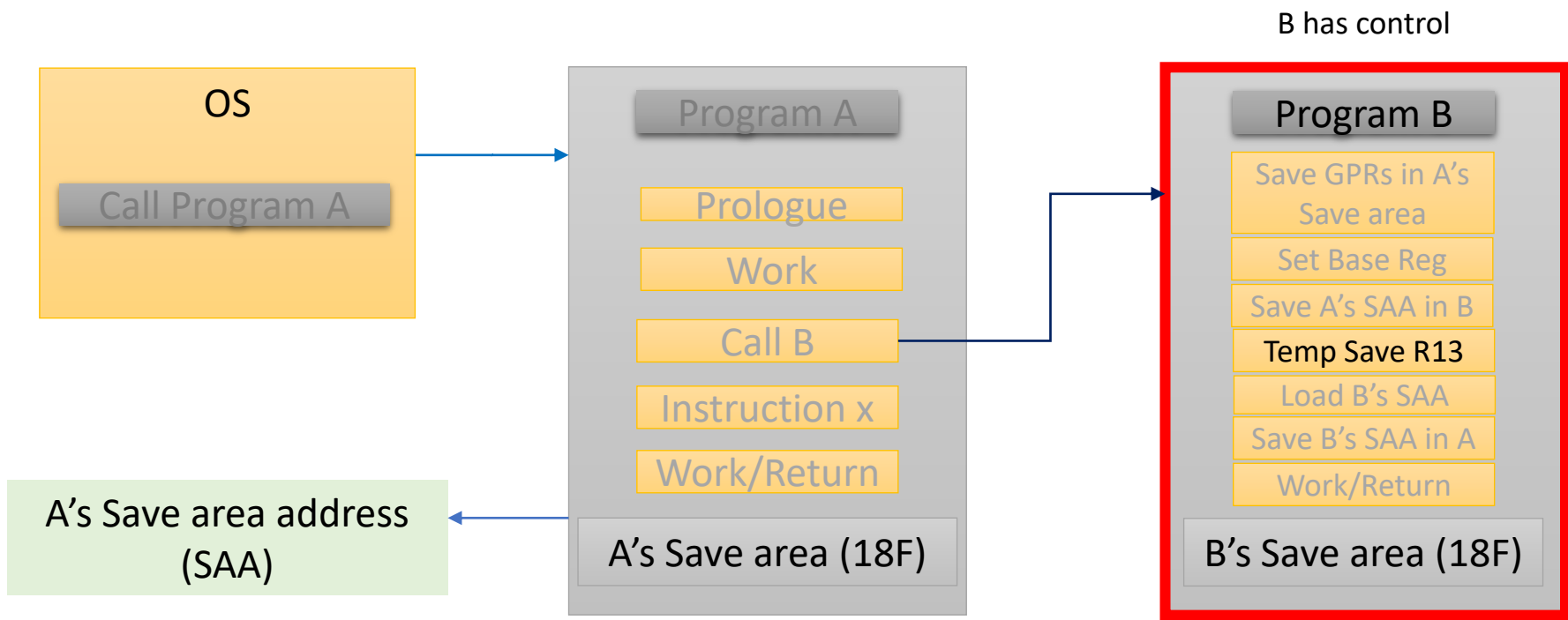
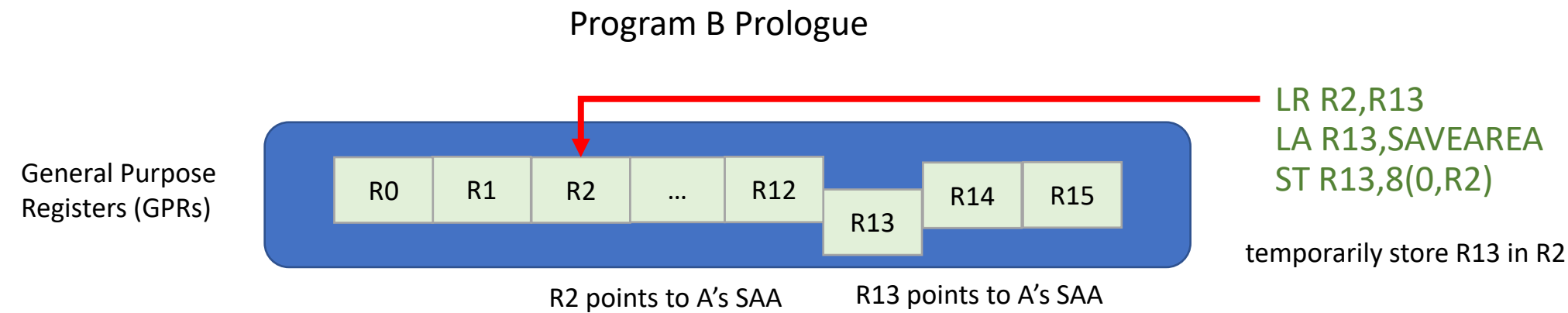
Establishing a base register with USING

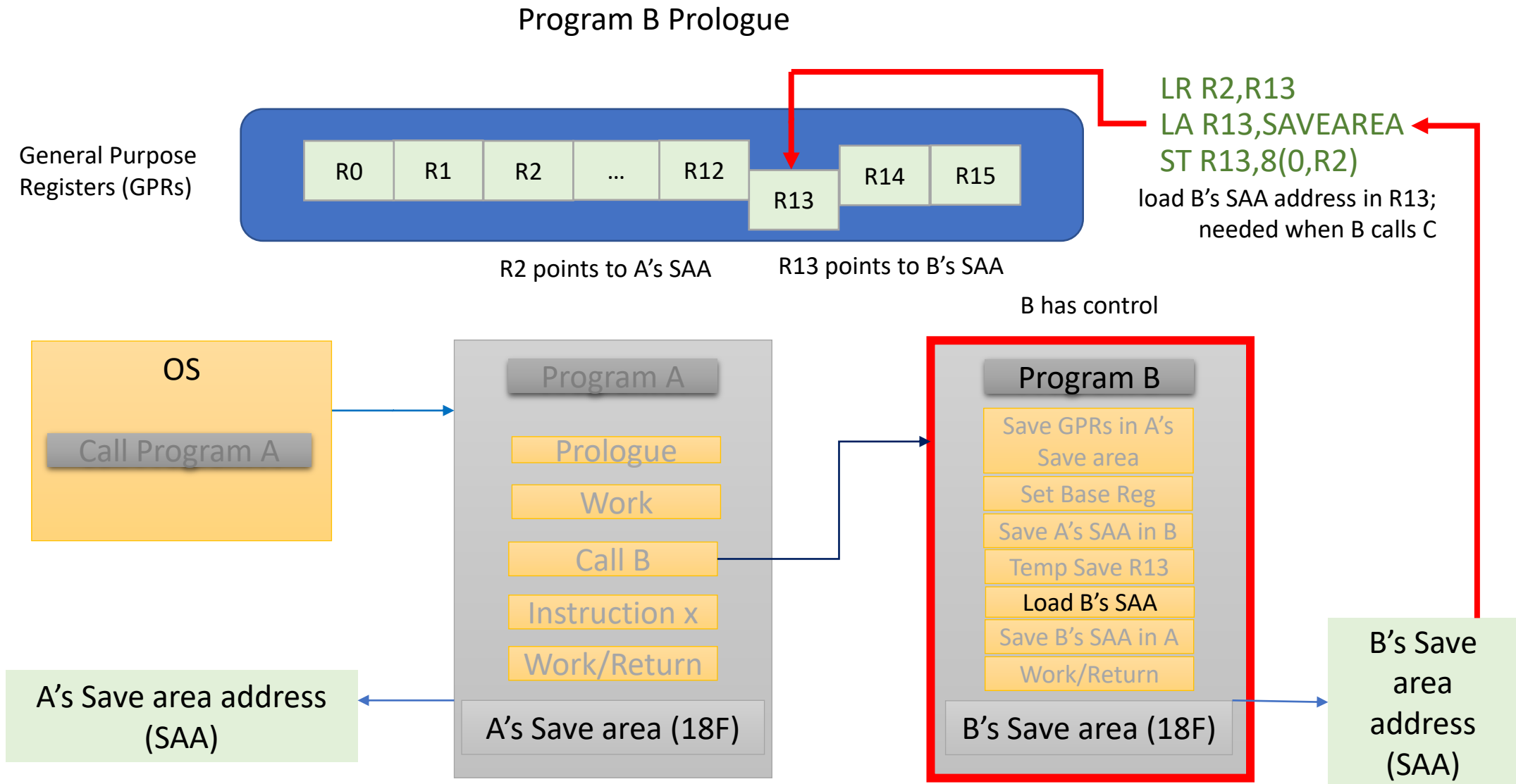
- USING instruction – tells the assembler which register is to be used as base and where to point relative to program start.

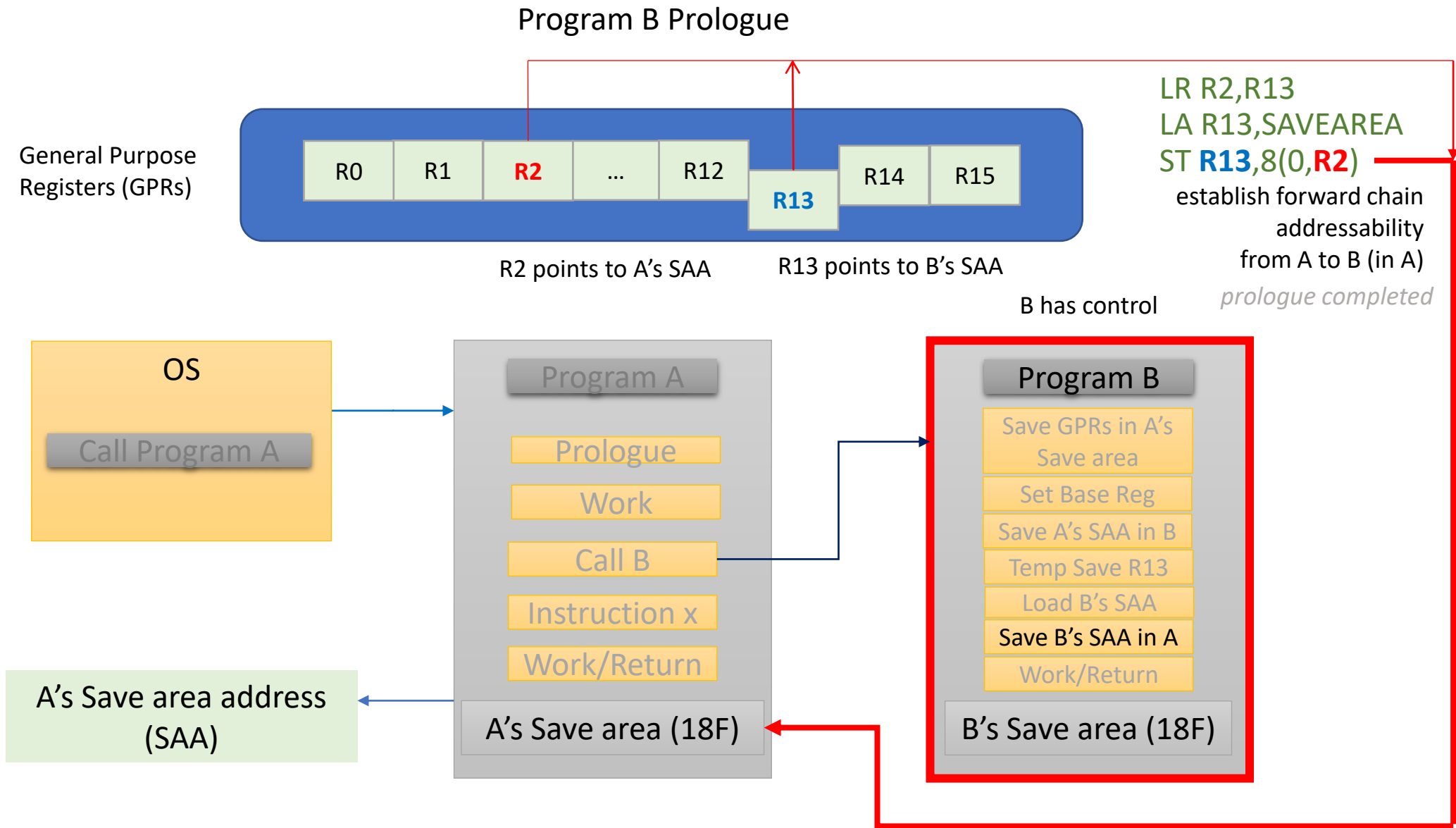
USING *,R12

- R12 is used as a typical base register. It contains an address set at execution time, depending on where the program is loaded in memory.



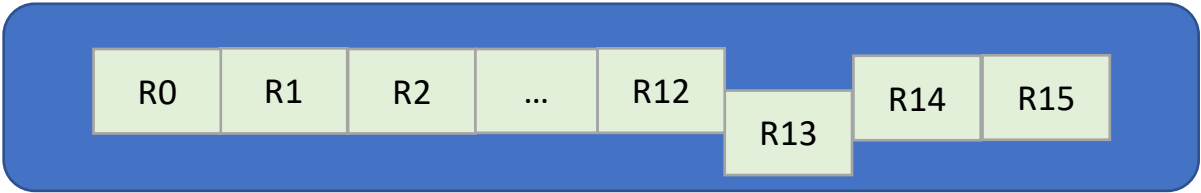




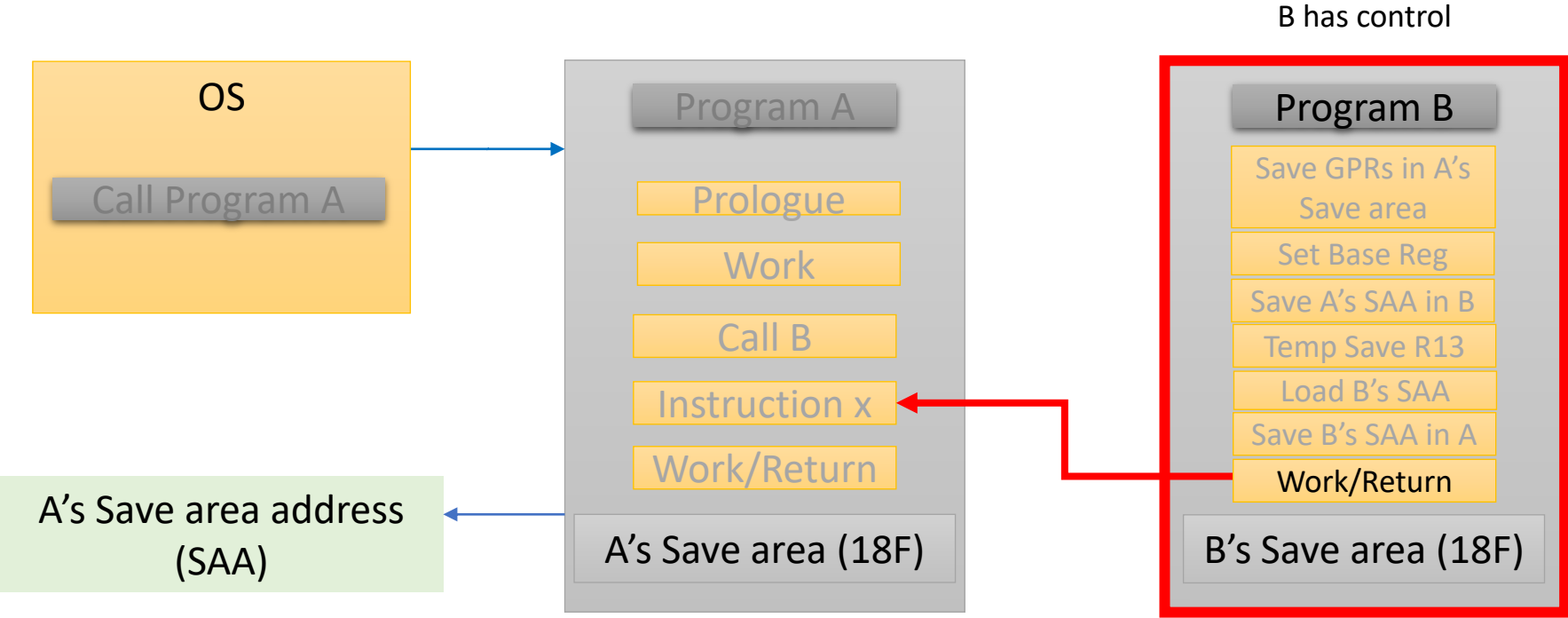


Program B Epilogue (i.e. Return)

General Purpose
Registers (GPRs)

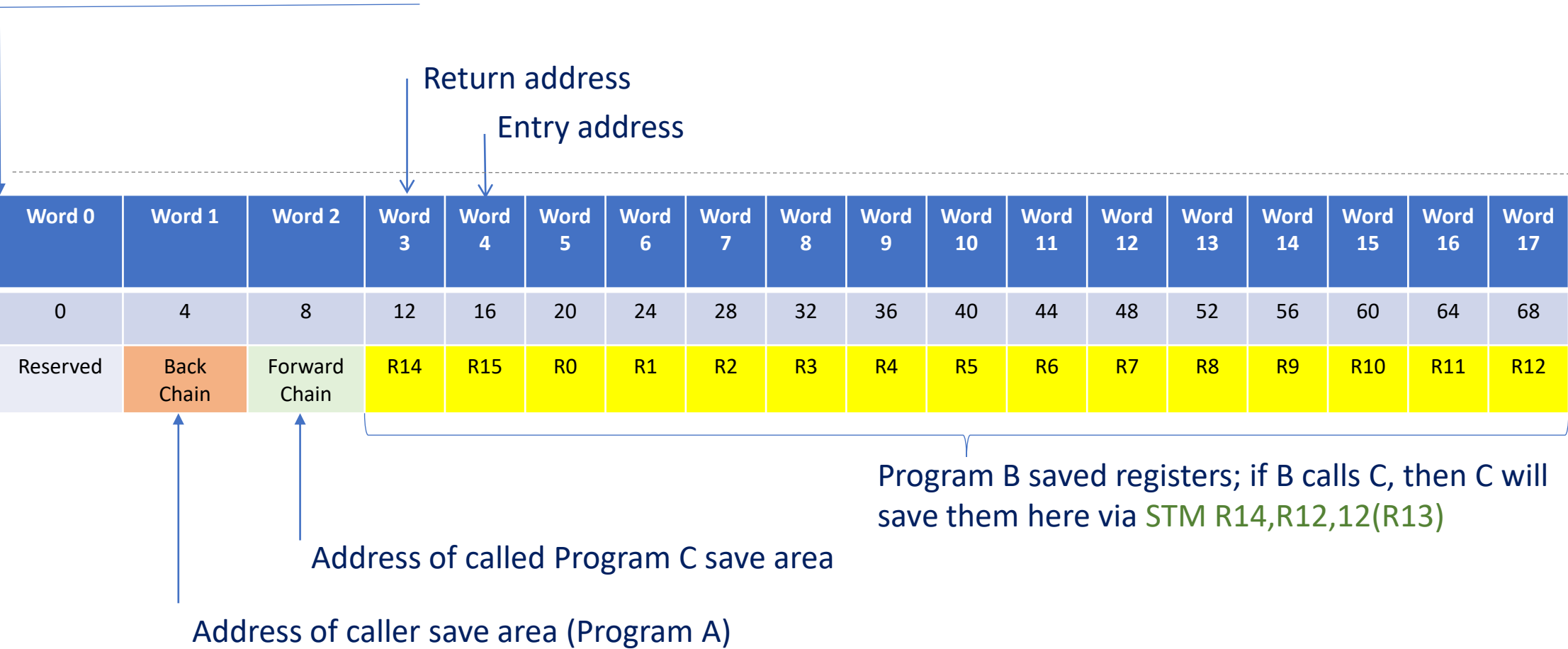


```
L    R13,SAVEAREA+4
LM   R14,R12,12(R13)
LA   R15,0
BR   R14
```



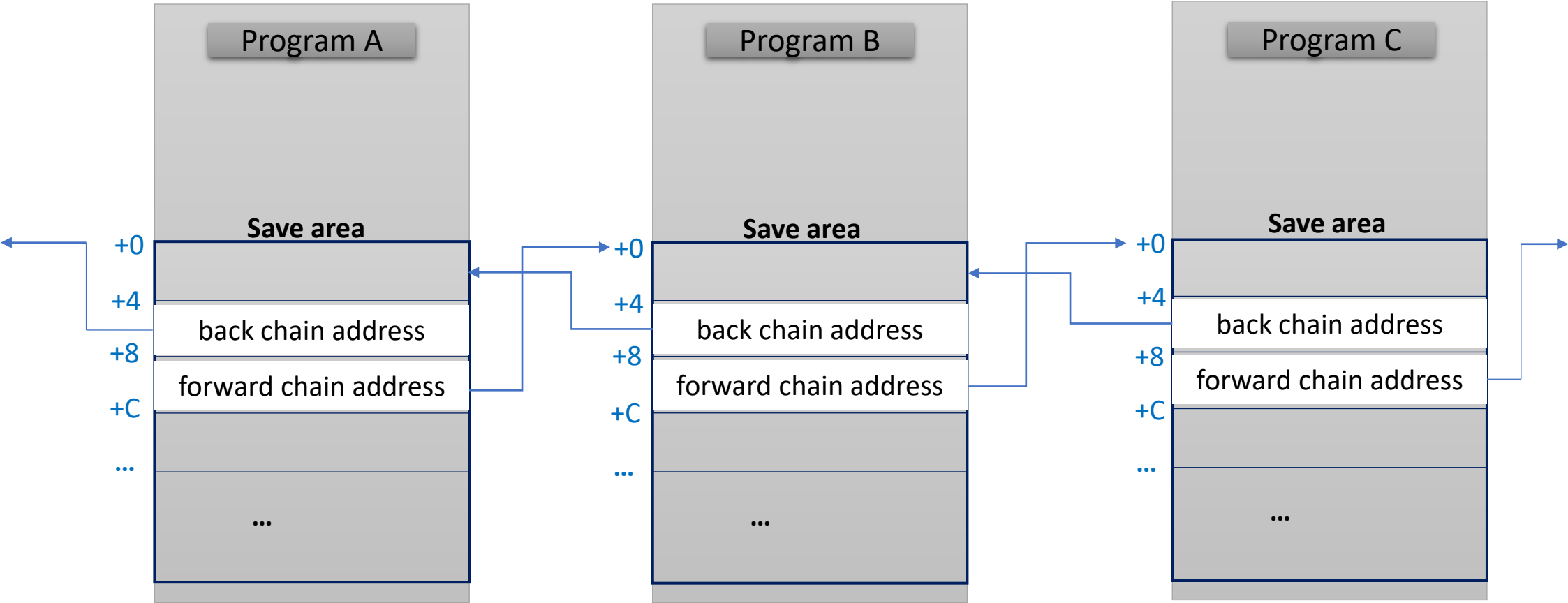
Save area in Program B

SAVEAREA DS 18F 18 fullwords (18x4 bytes = 72 bytes)



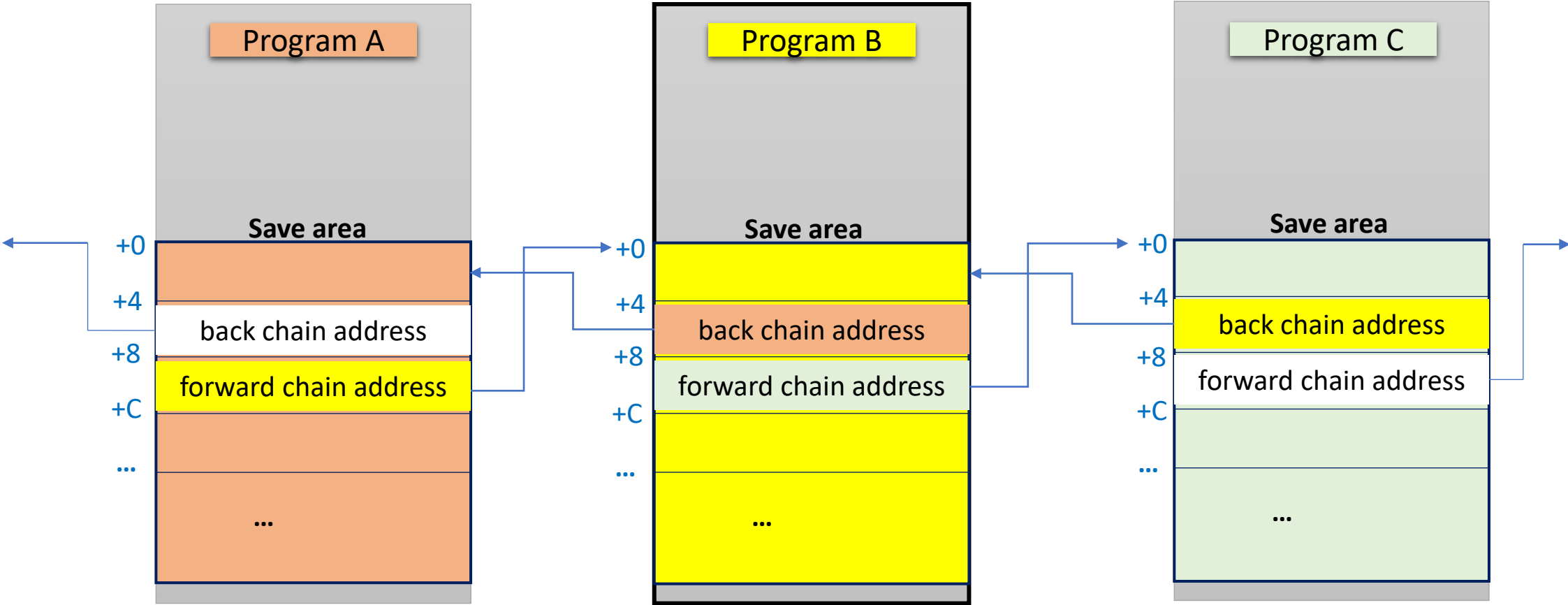
Chained save areas

Adapted from John Ehrman's book



Program B save area

Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11	Word 12	Word 13	Word 14	Word 15	Word 16	Word 17
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68
Reserved	Back Chain	Forward Chain	R14	R15	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12



Linkage – Prologue Code in Program B

PGMB	CSECT		
	STM	R14,R12,12(R13)	STM Store Multiple (a 32-bit instruction) STM saves only the low-order 32 bits of the GPRs STM takes registers R14, R15, R0, R1 ... R12 and stores them successively in 4 byte full words in caller's save area (address is R13) there is an offset of 12 bytes into the A's save area
	BASR	R12,R0	stores the address immediately following BASR in R12
	USING	*,R12	establish R12 as base register
	ST	R13,SAVEAREA+4	store caller's save area address in B's save area at Word 1 (back chain)
	LR	R2,R13	copy R13 to R2 temporarily
	LA	R13,SAVEAREA	load this program's save area address in R13 for calls to C
	ST	R13,8(0,R2)	store B's SAA in A's SAA at Word 2; this is forward chain from A to B

Linkage – Epilogue Code in Program B

...

instructions

work

...

L R13,SAVEAREA+4 retrieve address of caller's save area (Program A)

LM R14,R12,12(R13) restore registers

LA R15,0 set return code to 0

BR R14 return control to caller (Program A)

...

SAVEAREA DS 18F

LTORG *

END PGMB

Example using IBM’s Master The Mainframe Portal

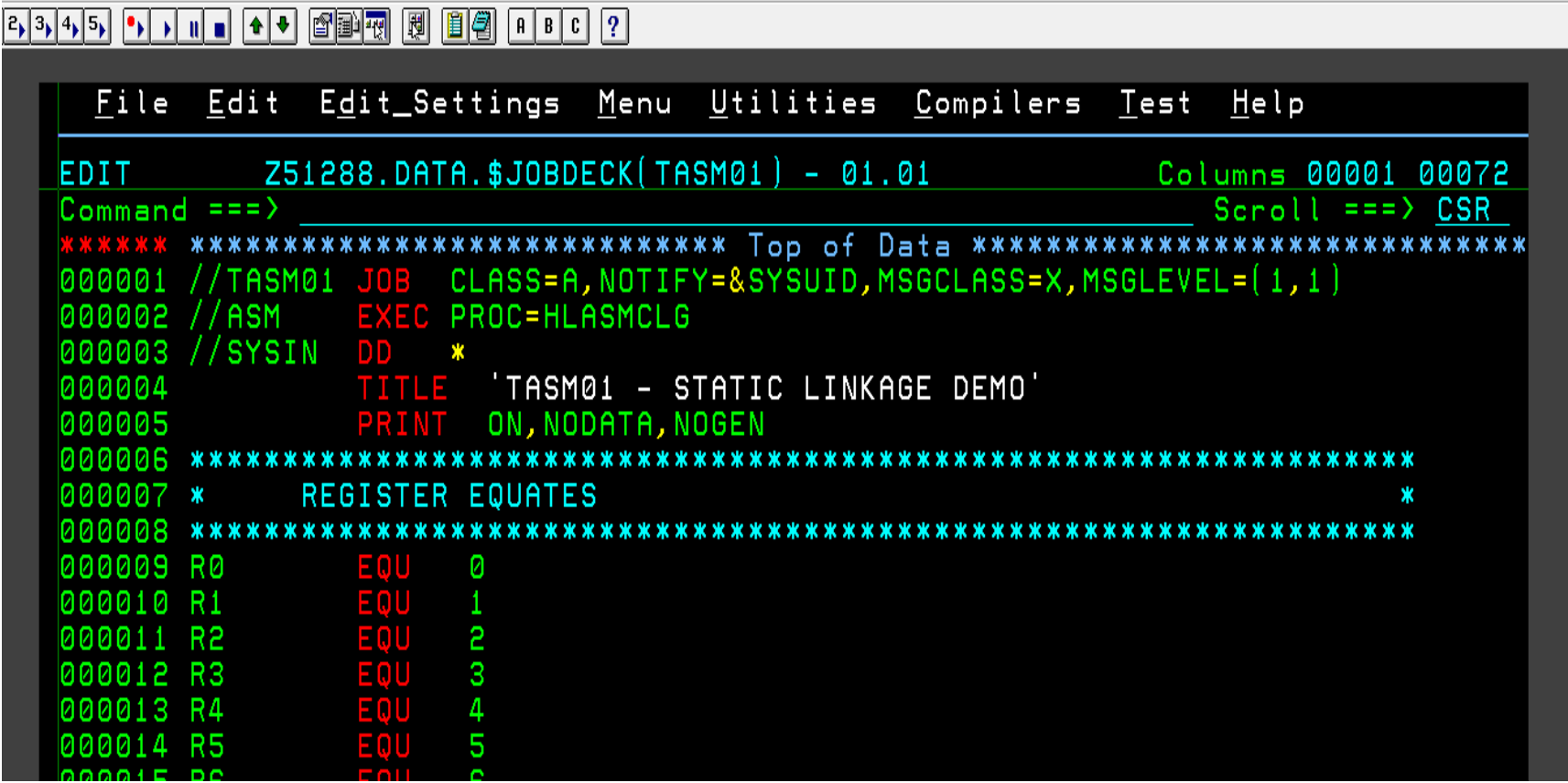
PDS	Z51288.DATA.\$JOBDECK
Member	TASM01

Job card info

Jobname	TASM01
Proc	HLASMCLG
Pgm Name	ASMT01
Output	G.PRINTER DD SYSOUT=*

Example using IBM's Master The Mainframe Portal

Job card



The screenshot shows a terminal window with a menu bar (File, Edit, Edit_Settings, Menu, Utilities, Compilers, Test, Help) and a toolbar. The main area displays a job card for a program named TASM01. The card includes standard JCL statements like //TASM01 JOB, //ASM EXEC, and //SYSIN DD, followed by a title and a list of register equates.

```
EDIT      Z51288.DATA.$JOBDECK(TASM01) - 01.01      Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** ***** Top of Data *****
000001 //TASM01 JOB  CLASS=A,NOTIFY=&SYSUID,MSGCLASS=X,MSGLEVEL=(1,1)
000002 //ASM      EXEC  PROC=HLASMCLG
000003 //SYSIN    DD   *
000004              TITLE  'TASM01 - STATIC LINKAGE DEMO'
000005              PRINT  ON,NODATA,NOGEN
000006 *****
000007 *          REGISTER EQUATES                               *
000008 *****
000009 R0          EQU    0
000010 R1          EQU    1
000011 R2          EQU    2
000012 R3          EQU    3
000013 R4          EQU    4
000014 R5          EQU    5
000015 R6          EQU    6
```

Prologue

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT Z51288.DATA.$JOBDECK(TASM01) - 01.02 Member TASM01 saved
Command ==> Scroll ==> CSR
000022 R13 EQU 13
000023 R14 EQU 14
000024 R15 EQU 15
000025 *****
000026 * SYMBOLIC EQUATES *
000027 *****
000028 SINGLE EQU X'09' WRITE AND SPACE 1
000029 DOUBLE EQU X'11' WRITE AND SPACE 2
000030 *****
000031 ASMT01 CSECT STANDARD ENTRY CODE
000032 STM R14,R12,12(R13) STORE REGS IN CALLER'S SAA
000033 BASR R12,R0 ESTABLISH R12 AS BASE REG
000034 USING *,R12
000035 ST R13,SAVEAREA+4 SAVE CALLER'S SAA IN CURRENT SAA
000036 LR R2,R13 COPY R13 TO R2 TEMPORARILY
000037 LA R13,SAVEAREA LOAD R13 WITH CURRENT SAA
000038 ST R13,8(R0,R2) STORE THIS SAA IN CALLER'S SAA
000039 *****
000040 * BEGIN THE PROGRAM LOGIC. FIRST OPEN THE INPUT AND OUTPUT FILES
000041 *****
000042 OPEN (PRINTER,(OUTPUT))
000043 MVI PRC1,DOUBLE PREPARE TO DOUBLE SPACE HEADER
000044 PUT PRINTER,PRHEAD PRINT THE HEADER
000045 MVI PRC2,DOUBLE PREPARE TO SINGLE SPACE REPORT
000046 PUT PRINTER,PRINTLN PRINT THE HEADER
000047 *
000048 * DISPLAY PACKED FIELDS
000049 MVI PRCLN1,DOUBLE DOUBLE SPACE LINE
000050 UNPK VALUE1(6),APK CONVERT APK FROM PACKED TO ZONED
000051 PUT PRINTER,PRINTLN1 PRINT LINE TYPE 1
000052 *
000053 MVI PRCLN1,SINGLE SINGLE SPACE LINE AFTERWARDS
000054 MVC VALUE1,VALUE1B BLANK OUT VALUE1
000055 UNPK VALUE1(3),BPK MOVE 2 BYTES TO VALUE1
000056 PUT PRINTER,PRINTLN1 PRINT LINE TYPE 1
000057 *
000058 * NOTE: DONT NEED A SINGLE SPACE MOVE TO PRINT LINE; DONE ABOVE
F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

SDSF
Job log

```

2 3 4 5
Display Filter View Print Options Search Help
-----
SDSF OUTPUT DISPLAY TASM01 JOB00459 DSID 2 LINE 0 COLUMNS 02- 81
COMMAND INPUT ==>
***** TOP OF DATA *****
JES2 JOB LOG -- SYSTEM SHOW 1 -- NODE

11.31.28 JOB00459 --- WEDNESDAY, 01 JAN 2020 ---
11.31.28 JOB00459 IRR010I USERID Z51288 IS ASSIGNED TO THIS JOB.
11.31.28 JOB00459 IEF677I WARNING MESSAGE(S) FOR JOB TASM01 ISSUED
11.31.28 JOB00459 ICH70001I Z51288 LAST ACCESS AT 11:28:40 ON WEDNESDAY, JANU
11.31.29 JOB00459 $HASP373 TASM01 STARTED - INIT 1 - CLASS A - SYS
11.31.29 JOB00459 - -----TIMINGS (MINS.)-----
11.31.29 JOB00459 -STEPNAME PROCSTEP RC EXCP CONN TCB SRB C
11.31.29 JOB00459 -ASM C 00 101 18 .00 .00
11.31.29 JOB00459 -ASM L 00 37 8 .00 .00
11.31.29 JOB00459 -ASM G 00 13 1 .00 .00
11.31.29 JOB00459 -TASM01 ENDED. NAME- TOTAL TCB CPU TIM
11.31.29 JOB00459 $HASP395 TASM01 ENDED - RC=0000
----- JES2 JOB STATISTICS -----
01 JAN 2020 JOB EXECUTION DATE
133 CARDS READ
597 SYSOUT PRINT RECORDS
0 SYSOUT PUNCH RECORDS
46 SYSOUT SPOOL KBYTES
0.01 MINUTES EXECUTION TIME
1 //TASM01 JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=X,MSGLEVEL=(1,1)
IEFC653I SUBSTITUTION JCL - CLASS=A,NOTIFY=Z51288,MSGCLASS=X,MSGLEVEL=
2 //ASM EXEC PROC=HLASMCLG
3 XXASMCLG PROC
XX*
XX*****
XX*
XX* Licensed Materials - Property of IBM
XX*
XX* 5696-234 5694-A01
XX*
XX* Copyright IBM Corporation 1992, 2008 All Rights Reserved.
XX*
XX* US Government Users Restricted Rights - Use, duplication
XX* or disclosure restricted by GSA ADP Schedule Contract
F1=HELP F2=SPLIT F3=END F4=RETURN F5=IFIND F6=BOOK
```

2 3 4 5 [Icons] [A] [B] [C] [?]

Display Filter View Print Options Search Help

SDSF STATUS DISPLAY ALL CLASSES LINE 1-2 (2)

COMMAND INPUT ==> SCROLL ==> CSR

NP	JOBNAME	JobID	Owner	Prt	Queue	C	Pos	SAff	ASys	Status
?	TASM01	JOB00459	Z51288	1	PRINT	A	1456			
	Z51288	TSU00457	Z51288	15	EXECUTION			S0W1	S0W1	

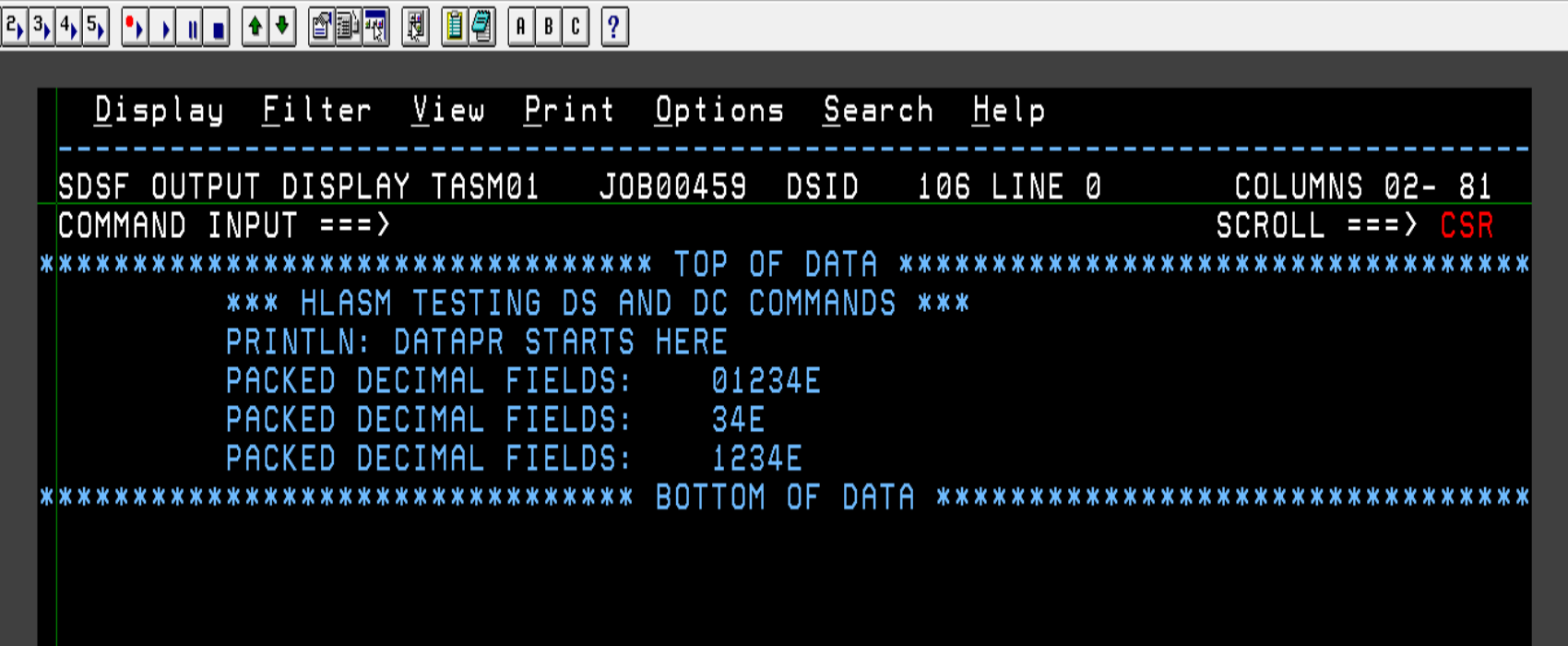
```

Display Filter View Print Options Search Help
-----
SDSF JOB DATA SET DISPLAY - JOB TASM01   (JOB00459)      LINE 1-6 (6)
COMMAND INPUT ==>                                          SCROLL ==> CSR
NP DDNAME StepName ProcStep DSID Owner C Dest Rec-Cnt Page
JESMSG LG JES2          2 Z51288 X LOCAL           21
JESJCL    JES2          3 Z51288 X LOCAL           53
JESYSMSG  JES2          4 Z51288 X LOCAL           72
SYSPRINT  ASM          C     102 Z51288 X LOCAL       285
SYSPRINT  ASM          L     103 Z51288 X LOCAL       153
S PRINTER ASM          G     106 Z51288 X LOCAL        5

```

SDSF Step Name

**SDSF
PRINTER
Output**



The screenshot shows a terminal window with a toolbar at the top containing icons for navigation and editing. The main display area has a menu bar with 'Display', 'Filter', 'View', 'Print', 'Options', 'Search', and 'Help'. Below the menu bar, a dashed line separates the header from the content. The header text reads: 'SDSF OUTPUT DISPLAY TASM01 JOB00459 DSID 106 LINE 0 COLUMNS 02- 81'. The first line of content is 'COMMAND INPUT ==>' followed by 'SCROLL ==> CSR' in red. The next line is a separator of asterisks: '***** TOP OF DATA *****'. This is followed by three lines of test results: '*** HLASM TESTING DS AND DC COMMANDS ***', 'PRINTLN: DATAPR STARTS HERE', and three lines of 'PACKED DECIMAL FIELDS:' with values '01234E', '34E', and '1234E'. The final line is another separator of asterisks: '***** BOTTOM OF DATA *****'.

```
Display Filter View Print Options Search Help
-----
SDSF OUTPUT DISPLAY TASM01 JOB00459 DSID 106 LINE 0 COLUMNS 02- 81
COMMAND INPUT ==> SCROLL ==> CSR
***** TOP OF DATA *****
*** HLASM TESTING DS AND DC COMMANDS ***
PRINTLN: DATAPR STARTS HERE
PACKED DECIMAL FIELDS: 01234E
PACKED DECIMAL FIELDS: 34E
PACKED DECIMAL FIELDS: 1234E
***** BOTTOM OF DATA *****
```

Summary

Static Linkage, non-reentrant, no recursion

24-bit or 31-bit addressing mode

Status preservation – ensuring nothing important is lost, modified, or destroyed in the process

Summary (cont.)

When working with standard linkage convention in z/OS, and dealing with 24- and 31-bit addressing mode (i.e. 32-bit registers), then

general registers 2 through 14 (R2-R14) must be saved by the called program (target) and restored to their original values before control is returned to the caller.

Next

Watch [Static Linkage Part II](#) - covers two key elements of calling conventions: control and argument passing.

Linkage – important topics

- 64-bit addressing mode & Format-4 save areas
- Program interaction in mixed more and Format-5 save areas
- Entry point identifiers
- Calling point identifiers
- Save area return flags
- Return codes
- Floating-point register conventions
- Assisted linkage
- Control
- Argument passing (variable length argument lists)

Presentation and JCL

https://github.com/MannyASM/HLASM_CallingConventions_StaticLinkage

Presentation	0001_HLASM_CallingConventions_StaticLinkage_I.pptx
JCL	ASM_TASM01_JOB_MTM.txt

OTHER REFERENCES

Redbooks

<http://www.redbooks.ibm.com/>

z/OS Library v2R4

<https://www-01.ibm.com/servers/resourceLink/svc00100.nsf/pages/zOSV2R4Library>

Manuals

z/OS ISPF User's Guide Vol I, z/OS ISPF User's Guide Vol II

z/OS MVS JCL User's Guide

z/OS SDSF User's Guide

z/OS MVS Data Areas Volume 1 (ABE - IAR)

z/OS MVS Data Areas Volume 2 (IAX - ISG)

z/OS MVS Data Areas Volume 3 (ITK - RQE)

z/OS MVS Data Areas Volume 4 (RRP - XTL)

Moshix Mainframe Channel

<https://www.youtube.com/channel/UCR1ajTWGiUtiAv8X-hpBY7w>

OTHER REFERENCES (cont.)

IDCP – Institute for Data Center Professionals

http://idcp.marist.edu/enterprisesystemseducation/zos_program_overview/assemblerprogrammingcertificate.html

Northern Illinois University

<http://faculty.cs.niu.edu/~byrnes/csci360/notes.html>

IBM Master The Mainframe

<https://masterthemainframe.com/> (students)

<https://www-01.ibm.com/events/wwc/ast/mtm/audit.nsf/enrollall> (non-students)