Embedded Software development Process and Tools:

Lesson-4
Linking and Locating Software

1. Linker

Linker

- Links the compiled codes of application software, object codes from library and OS kernel functions.
- Linking necessitates because of the number of codes to be linked for the final binary file.

Linking Necessity

- For example, standard codes for to program a delay task, must link with the assembled codes.
- The delay code sequential from a beginning address.
- The assembly software codes also sequential from another beginning address.
- Both the codes have to at the distinct addresses as well as at the available addresses in the system. Linker links these

Linked binary file

- After linking, re-allocation of the sequences of placing the codes before actually placement of the codes in the memory
- Linked file in binary for *run* on a computer commonly known as executable file or simply '.exe' file.

2. Loader

Loader

- Program loaded in a computer RAM.
- Loader program performs the task of *reallocating* the codes after finding the physical memory addresses available at a given instant

Loader…

- Loader a part of the operating system and places codes into the memory after reading the '.exe' file.
- Step necessary because the available memory addresses may not start from 0x0000, and binary codes have to be loaded at the different addresses during the run.

Loader…

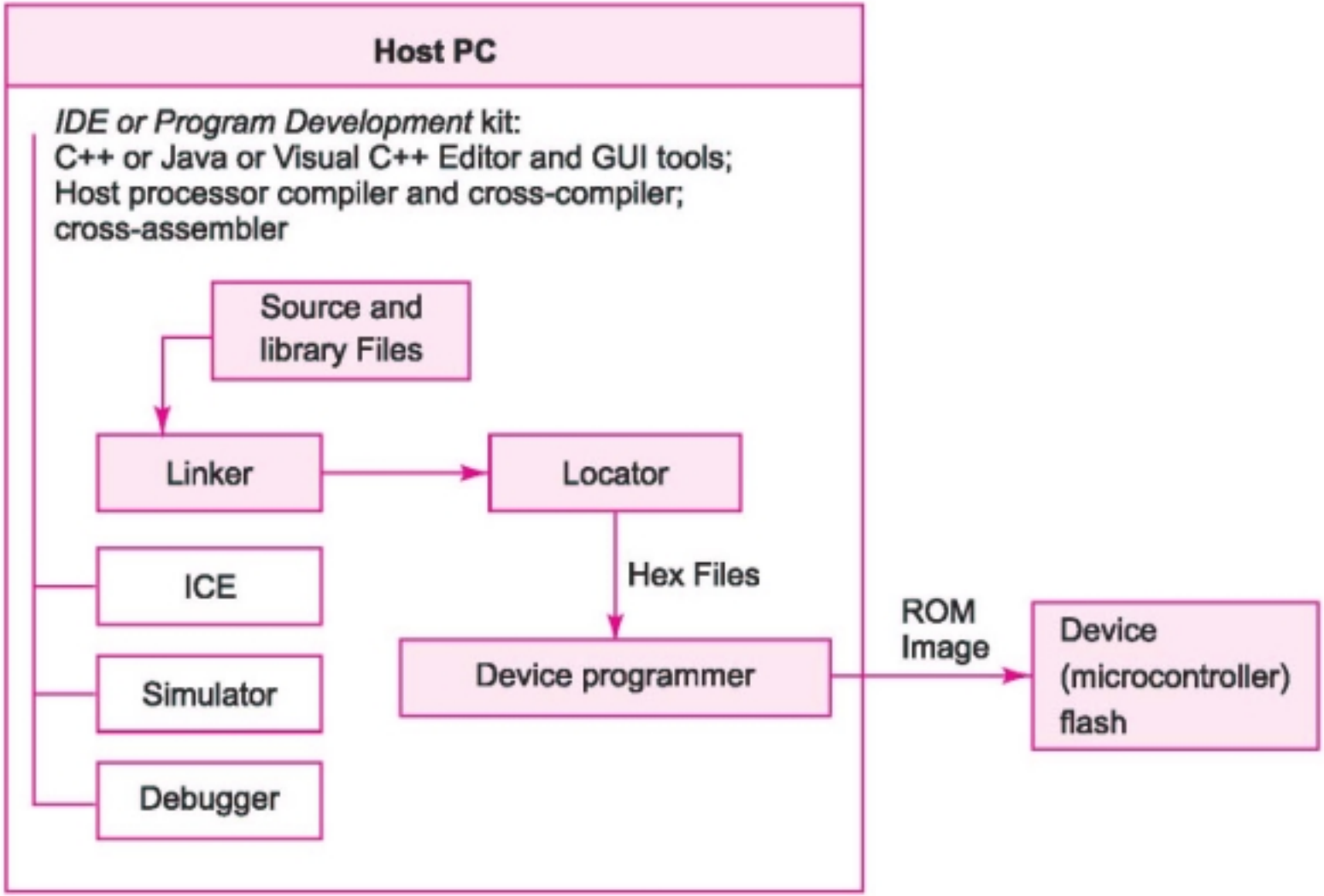
- Loader finds the appropriate start address.
- In a computer, after the loader loads into a section of RAM and after loading the program ready to run

3. Locator

Locator

- When the code embeds into ROM or flash, a system design process is *locating* these codes as a ROM image.
- Codes are permanently placed at the actually available addresses in the ROM.
- Embedded systems─ no separate program to keep track of the available addresses at different times during the running, as in a computer .

Various software tools and chain of actions of linker at host and locator in an embedded system



Locator…

- Next step after linking─ use of a locator for the program codes and data in place of loader

Locator…

- The locator is specified by the programmer as per available addresses at the RAM and ROM in target.
- Programmer defines the available addresses in embedded systems for loading to load and creating files for permanently locating the codes using a device programmer

Locator…

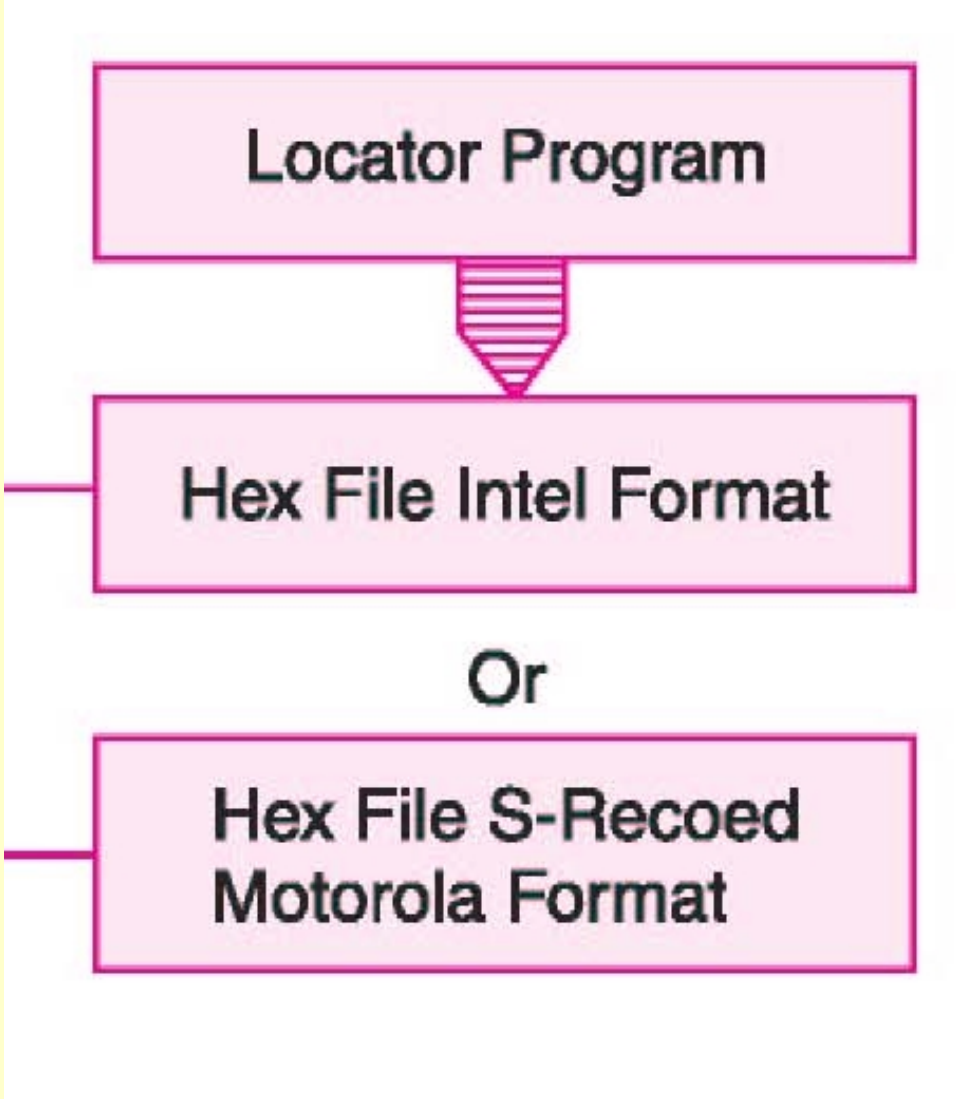
- Uses cross-assembler output, a memory allocation map and provides the locator program output file.
- Locator program output is in the Intel hex file or Motorola S- record format.

Locator…

- Uses the cross compile codes in different cross-compiled segments for (i) instructions, (ii) initialized values and addresses (iii) constant strings (iv) un-initialized data.
- Locates the I/O tasks and hardware device driver codes at the unchanged addresses. These are as per the interfacing circuit between the system buses and ports or devices.

4. Locator Output in Intel hex file or Motorola S- record format

Locator Output



Locator Output in Intel hex file

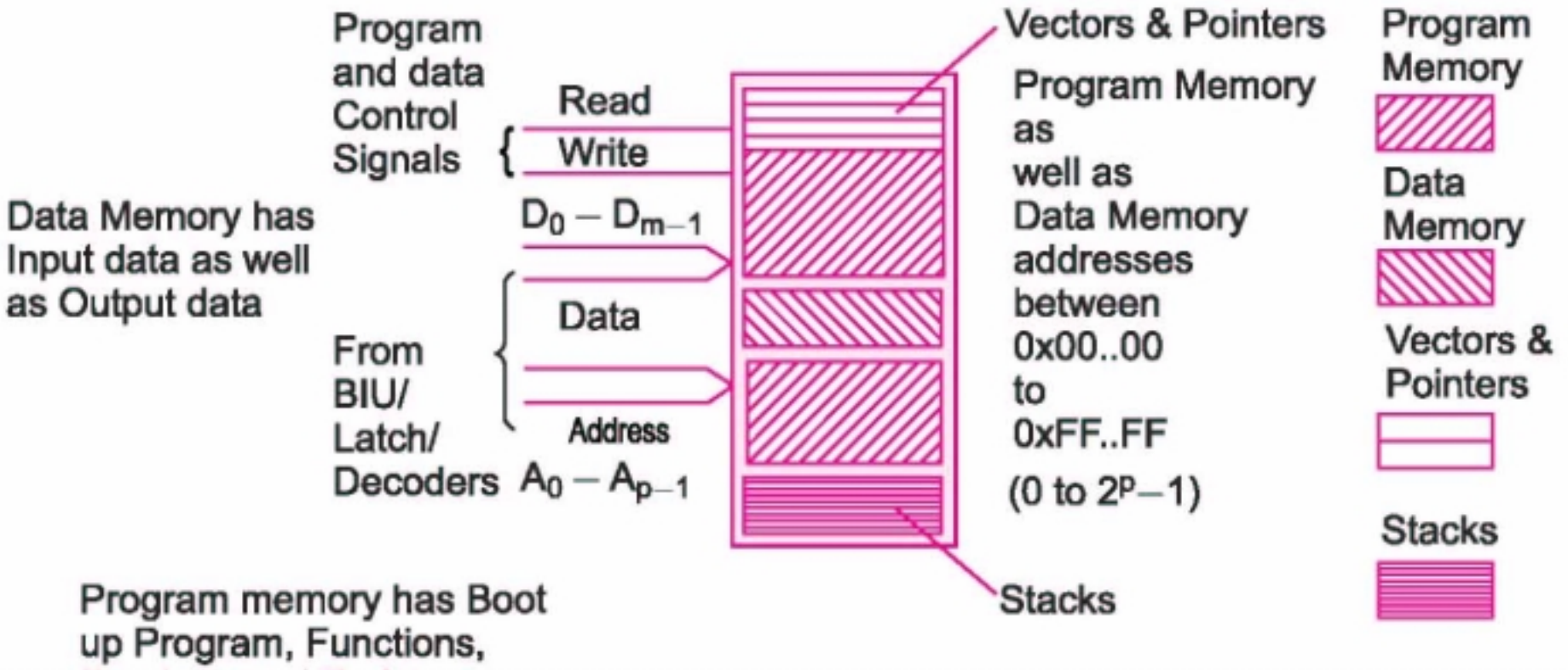
Line Number[1]	*First Character*	*Second and Third Characters for C*[2]		*Address, Addr*[3]	*Sixth and Seventh Characters*[4]		*N_d*[5] *Bytes for Storage in ROM from Addr (Maximum value of N_d can be 253 decimal)*	*Check-sum*[5]
0	:	0	C	0000	0	0	aa bb cc dd ee ff xx yy zz bb cc dd	cs0
1	:	0	8	000C	0	0	cc aa cc dd ee ff xx yy	cs1
2	:	0	E	0014	0	0	dd bb cc dd ee ff xx yy zz bb cc dd aa xx	cs2
3	:	0	1	0022	0	0	0A	cs3
4	:	0	4	0023	0	0	dd bb cc dd	cs4
5	:	1	0.	0027	0	0	dd bb cc dd ee ff xx yy zz bb cc dd aa ff 01 c0	cs5

Locator Output in Motorola S- record format

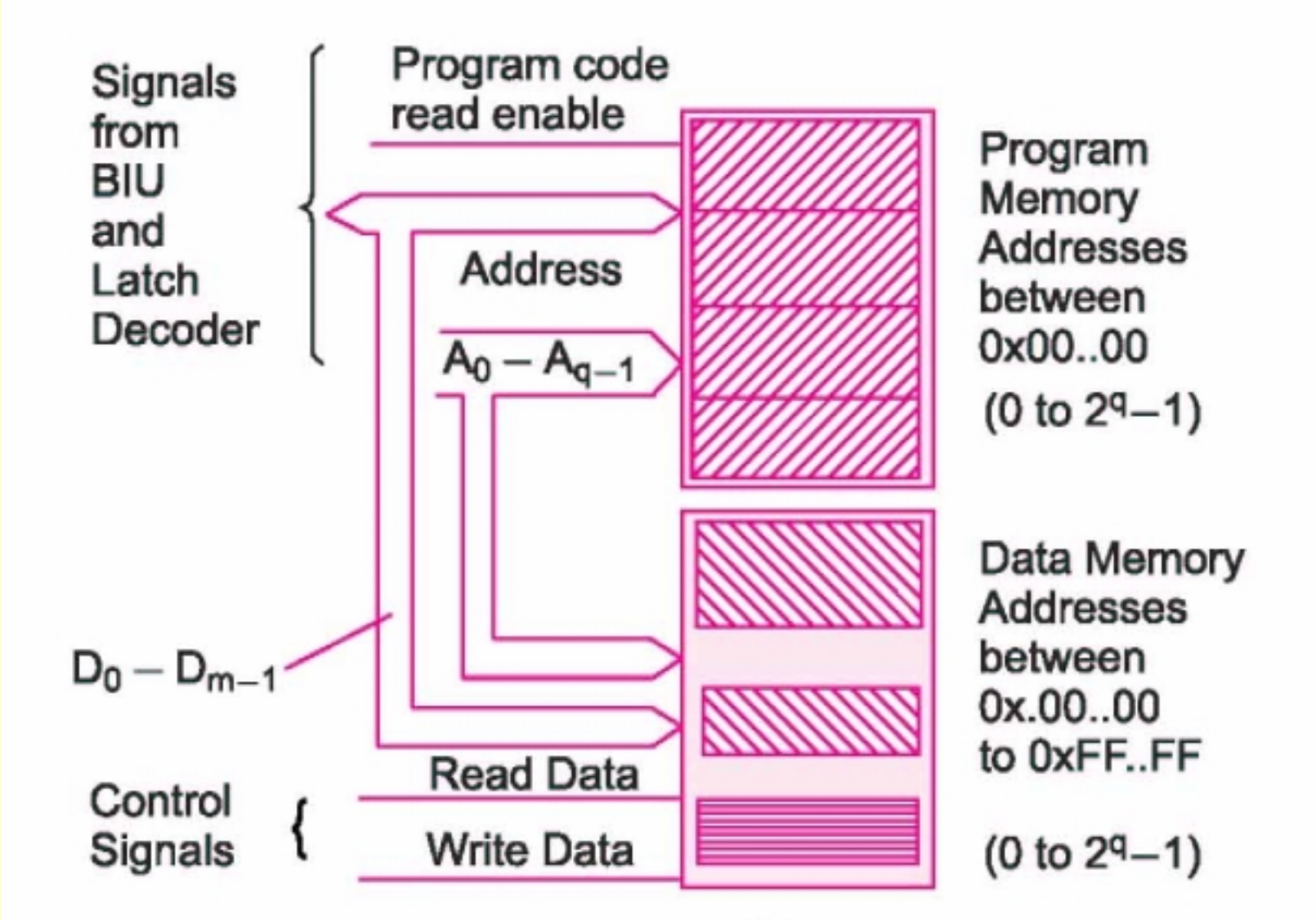
Line Number[1]	*First Character*	*Second Character*[2]	*Third and Fourth Characters for N*[3]		*Address, Addr*[4]	*N_d*[$] *Bytes for Storage in ROM from Addr (Maximum value of N_d can be 253 decimal)*	*Check-sum*[5]
0	S	2	1	0	000000	aa bb cc dd ee ff xx yy zz bb cc dd	cs0
1	S	2	0	C	00000C	cc aa cc dd ee ff xx yy	cs1
2	S	2	1	2	000014	dd bb cc dd ee ff xx yy zz bb cc dd aa xx	cs2
3	S	2	0	5	000022	0A	cs3
4	S	2	0	8	000023	dd bb cc dd	cs4
5	S	2	1	4	000027	dd bb cc dd ee ff xx yy zz bb cc dd aa ff 01 c0	cs5

5. Memory Map for coding a locator program

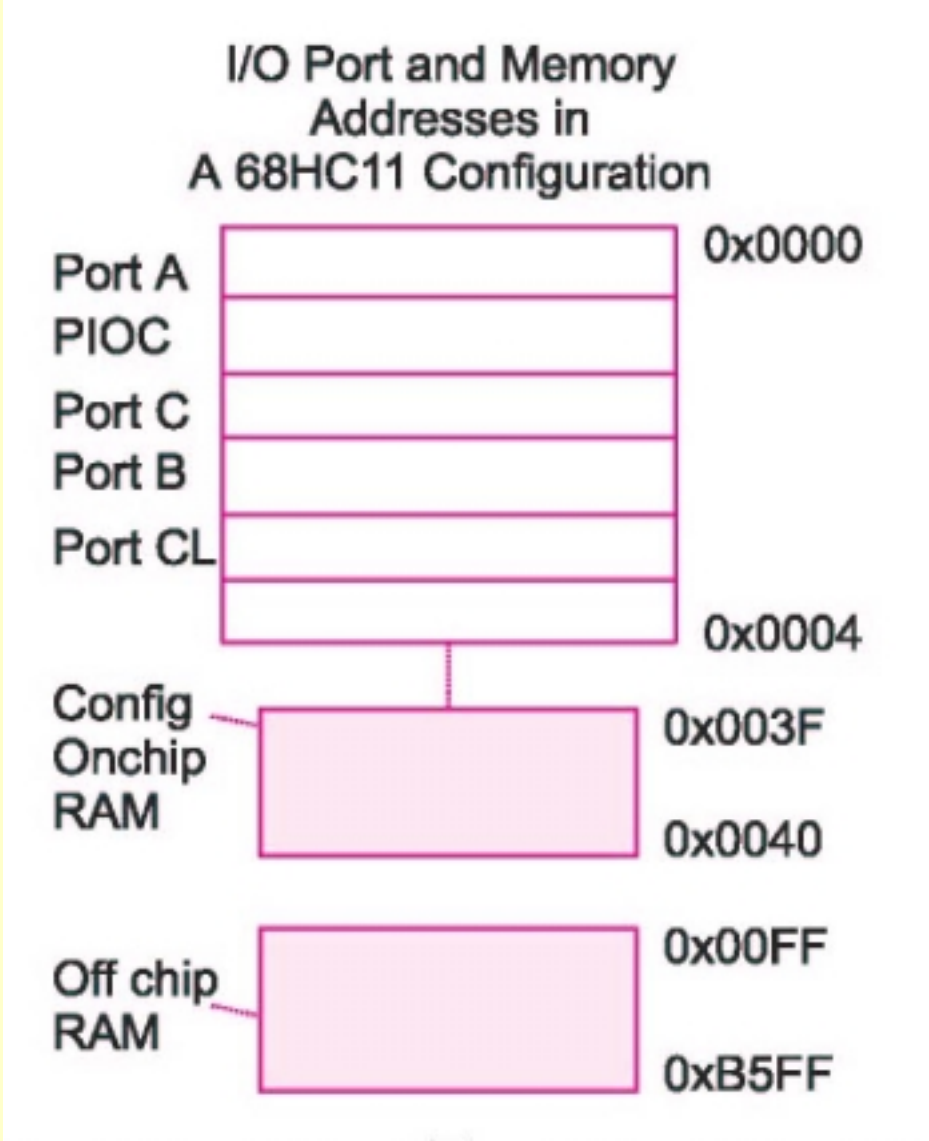
Memory map in Princeton Architecture



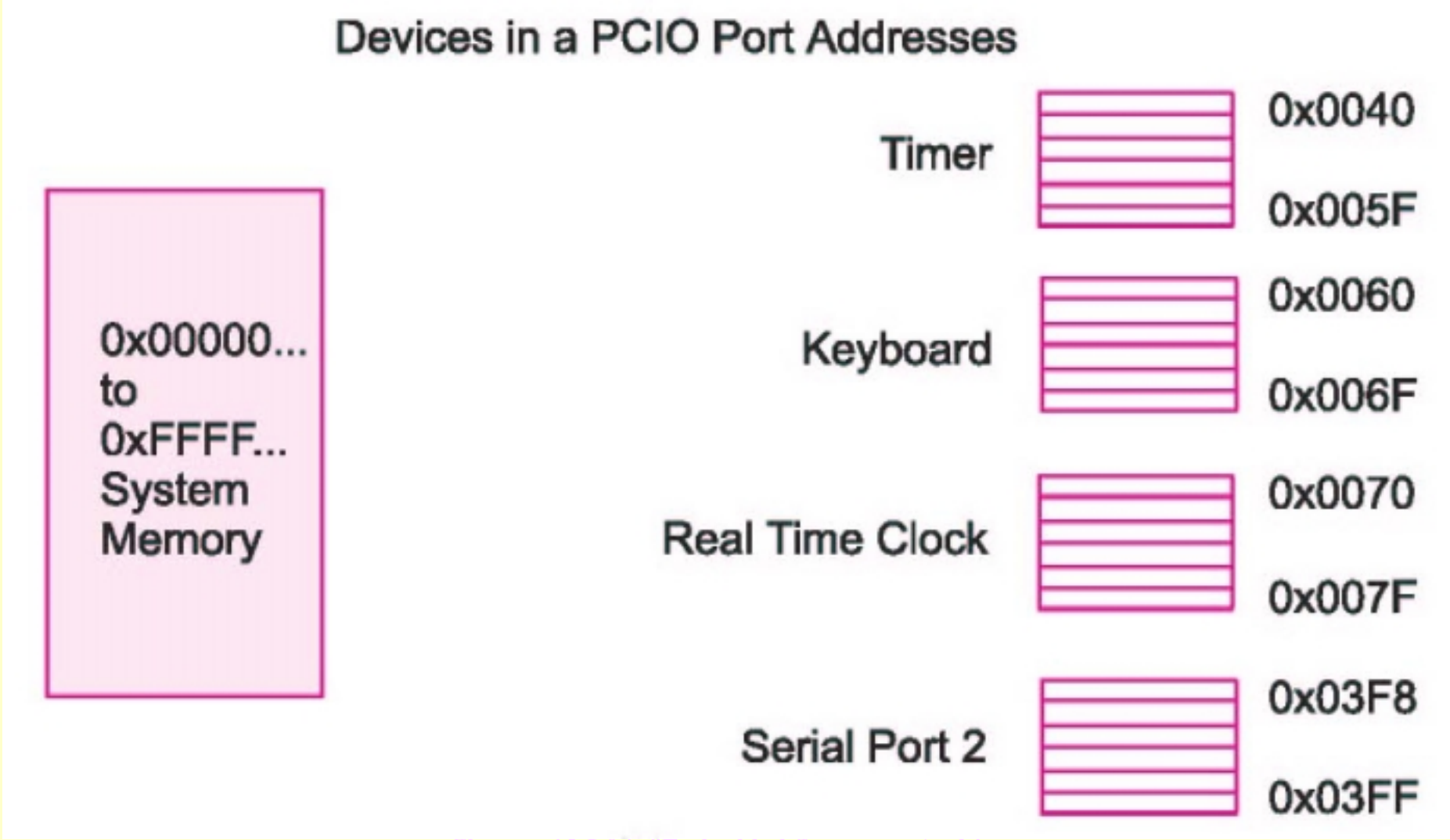
Memory map in Harvard Architecture



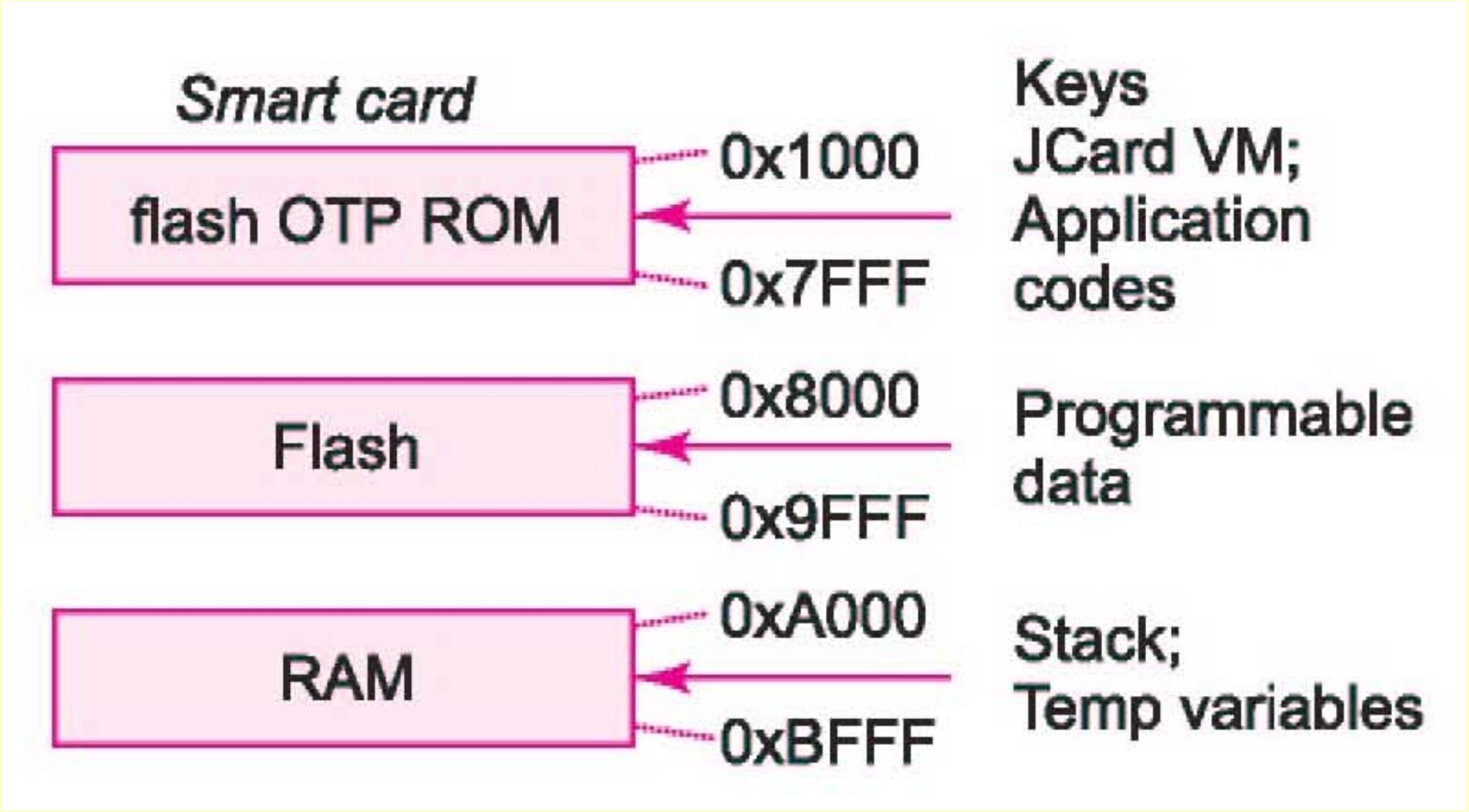
IO port, memory and devices address spaces in 68HC11



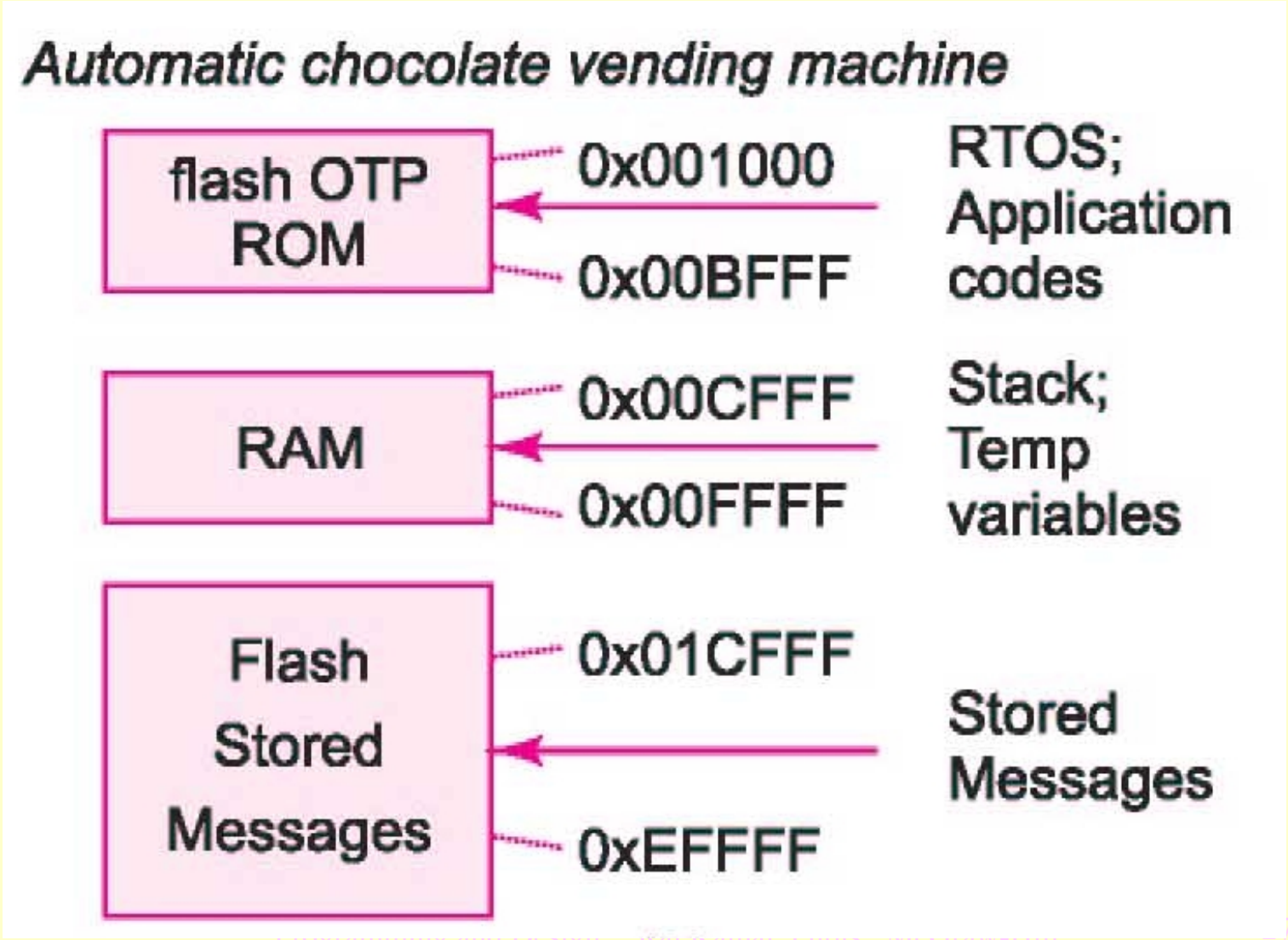
Device Addresses in 80x86-based host system



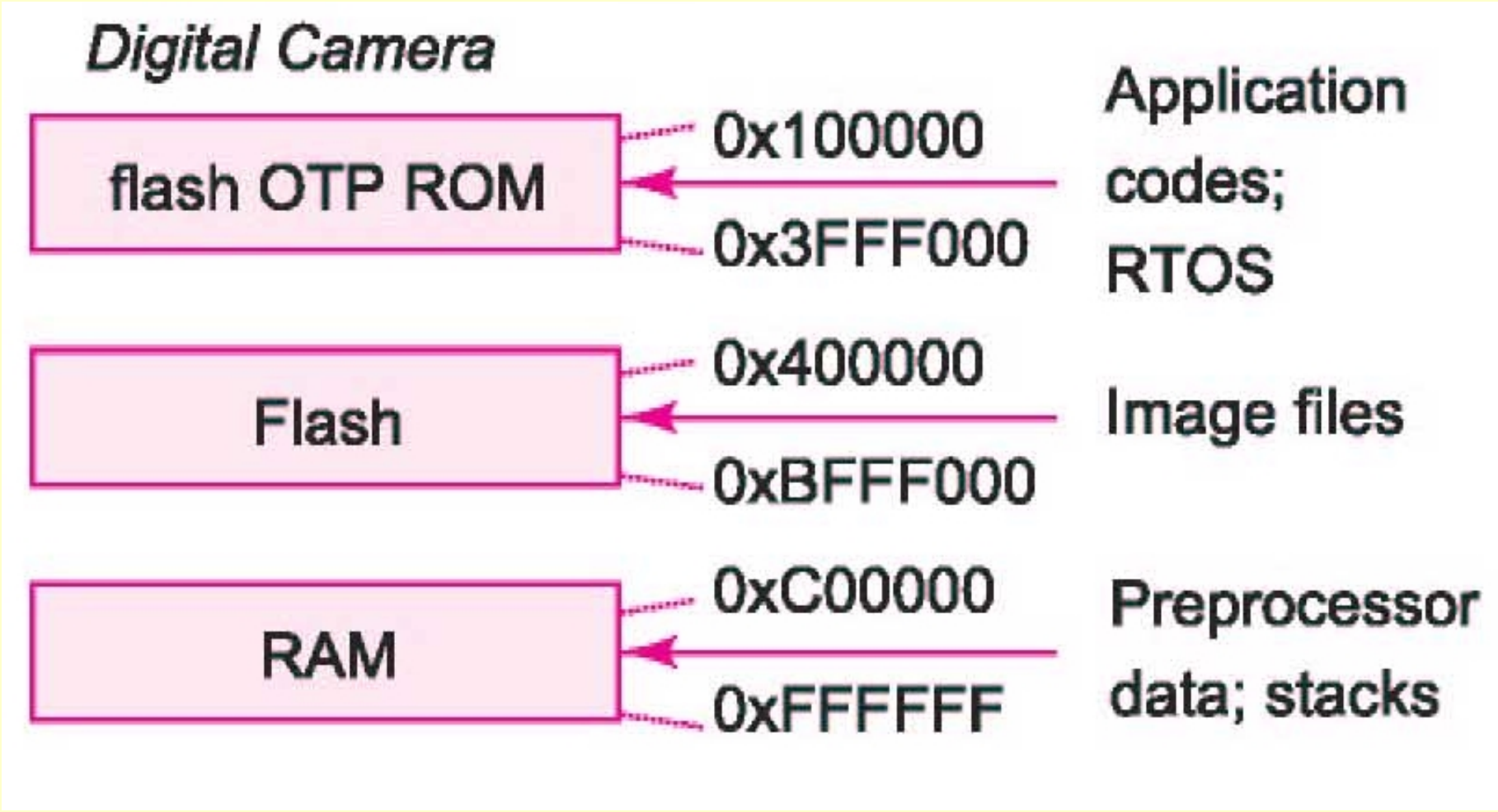
A smart card system memory allocation map for the Locator program



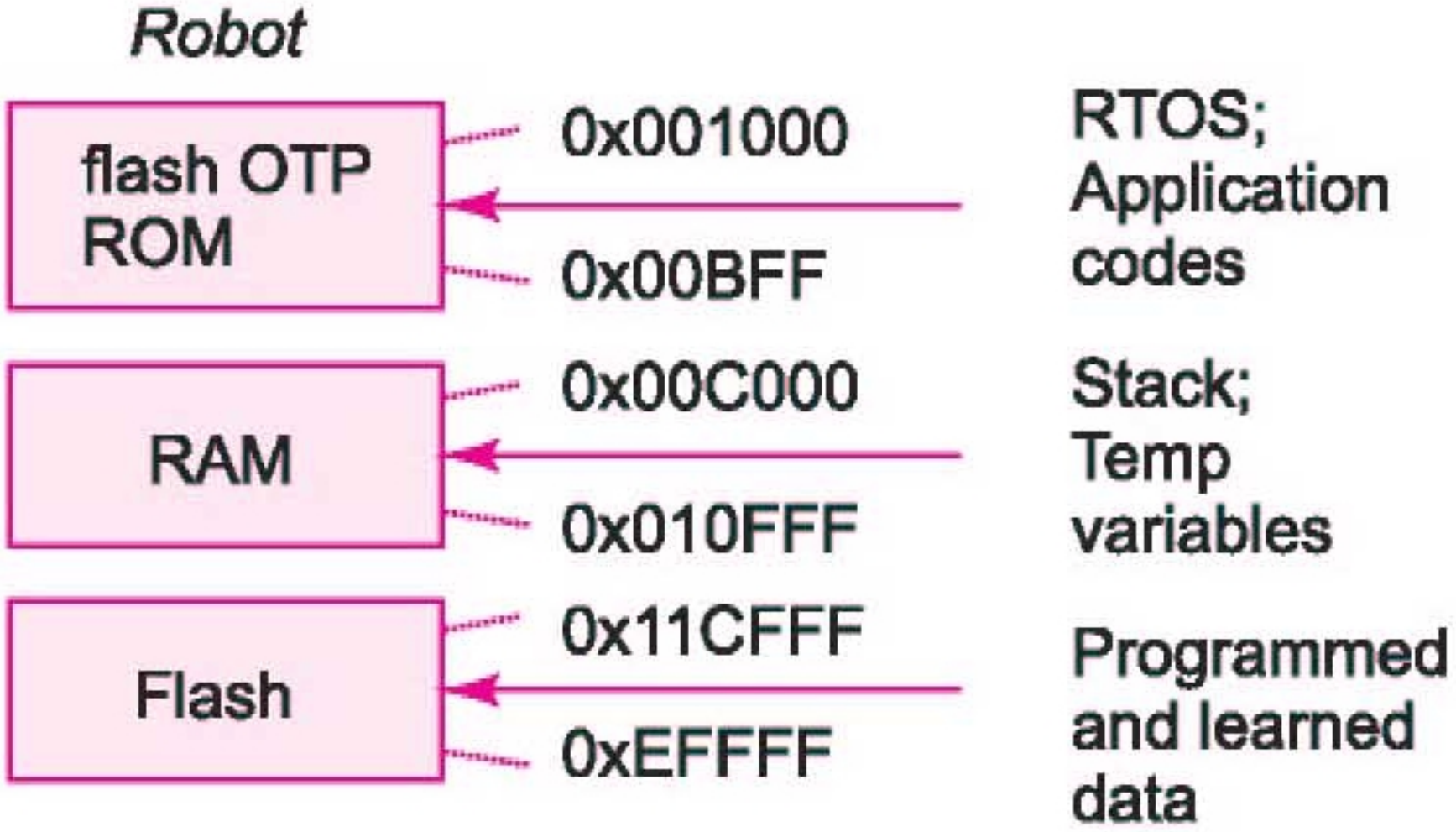
An automatic chocolate vending machine memory allocation for the Locator program



A digital camera system memory allocation map for the Locator program



A robot system memory allocation map for the Locator programs



Summary

We learnt

- Linker and locator used for developing the codes for the target hardware
- Locator files in Intel Hex or Motorola S format.
- Main memory Harvard architecture, the program memory map separate
- Main memory Princeton architecture, the program and data memory map same

We learnt

- Memory map used for coding locator software
- Memory map defined for a locator includes the device I/O addresses, designed after appropriate address allocations of the pointers, vectors, data sets, and data structures.

We learnt

- Memory map used for coding locator software.
- Memory map defined for a locator includes the device I/O addresses
- Map designed after appropriate address allocations of the pointers, vectors, data sets, and data structures
-

End of Lesson-4 of chapter 13 on Linking and Locating Software