Input/Output

- May be specific instructions
- May be done using data movement instructions (memory mapped)
- May be done by a separate controller (DMA)

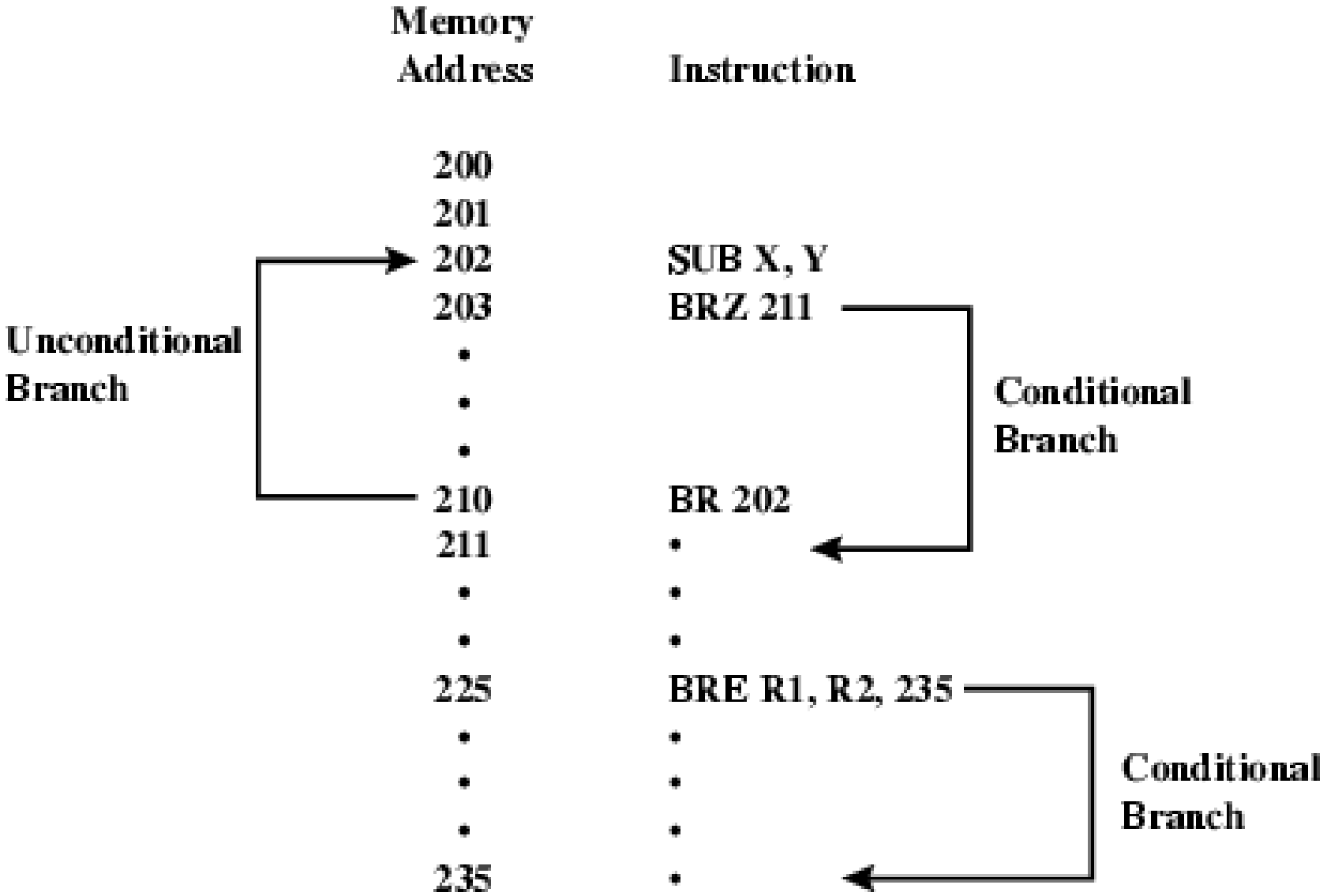
Systems Control

- Privileged instructions
- CPU needs to be in specific state
  - Ring 0 on 80386+
  - Kernel mode
- For operating systems use

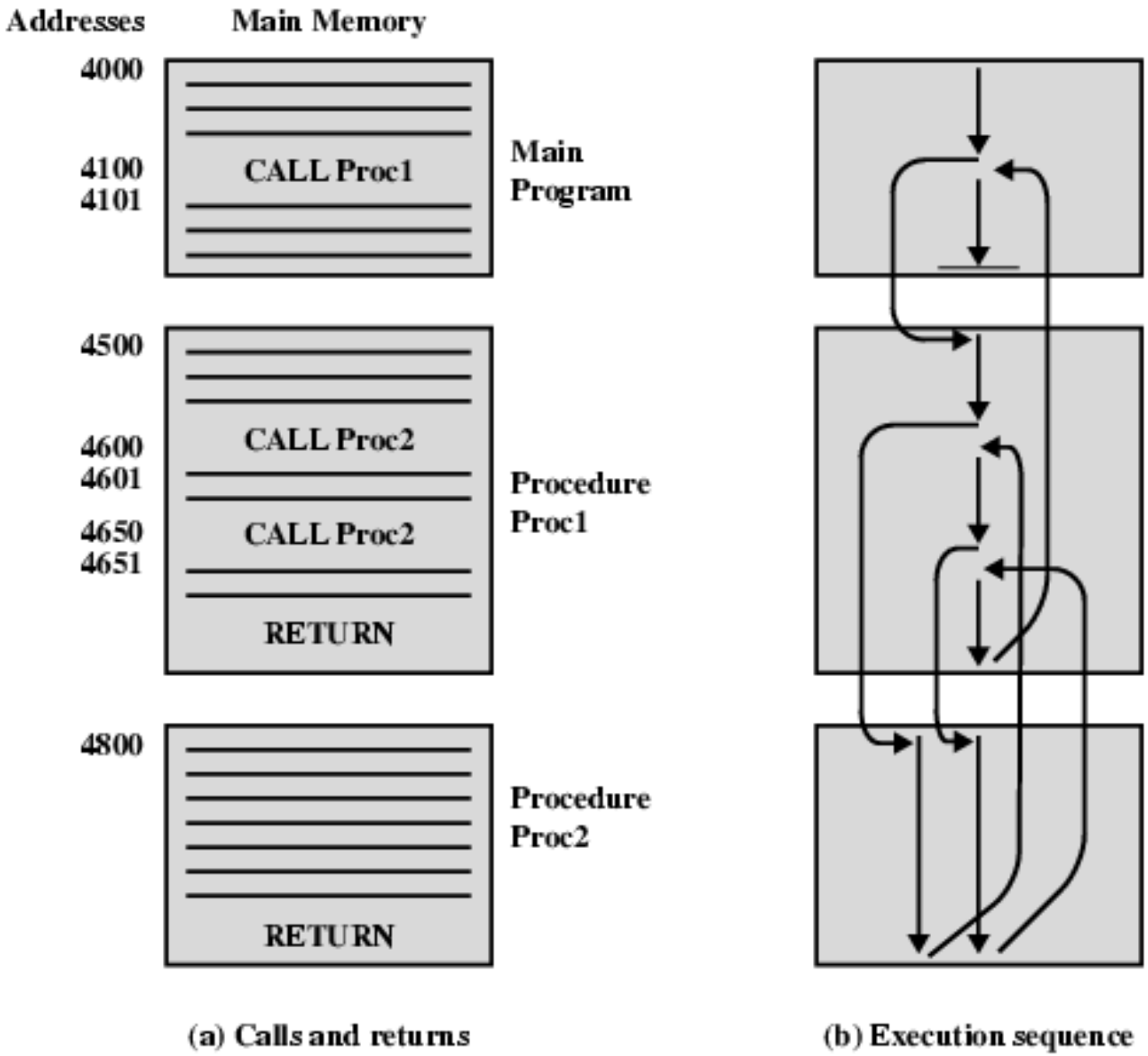
Transfer of Control

- Branch
  - —e.g. branch to x if result is zero
- Skip
  - —e.g. increment and skip if zero
  - —ISZ Register1
  - —Branch xxxx
  - —ADD A
- Subroutine call
  - —c.f. interrupt call

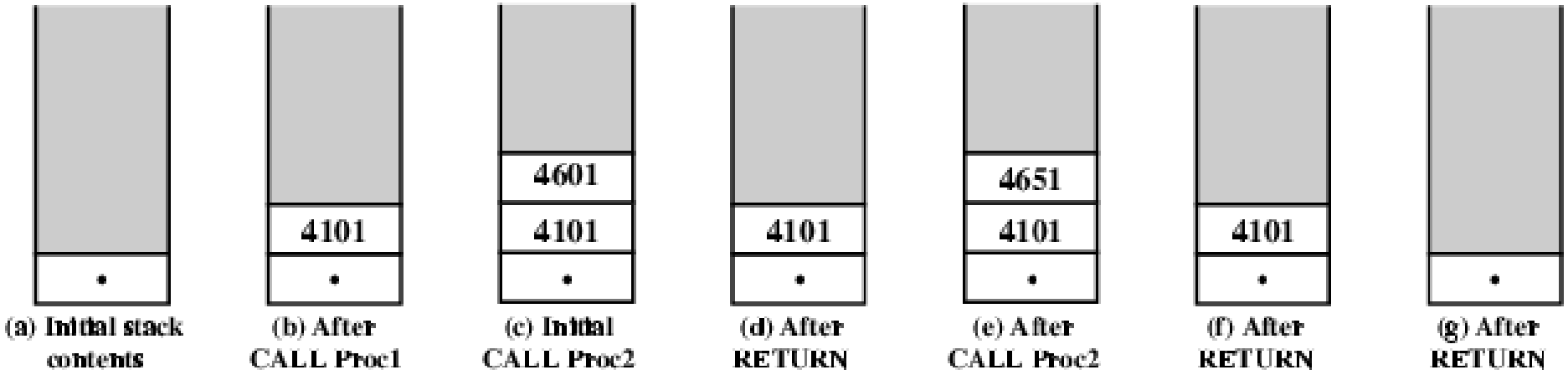
Branch Instruction



Nested Procedure Calls



Use of Stack


4601
4101
4101
4651
4101
4101
4101
(a) Initial stack contents
(b) After CALL Proc1
(c) Initial CALL Proc2
(d) After RETURN
(e) After CALL Proc2
(f) After RETURN
(g) After RETURN

Exercise For Reader

- Find out about instruction set for Pentium and PowerPC
- Start with Stallings
- Visit web sites

Byte Order
(A portion of chips?)

- What order do we read numbers that occupy more than one byte
- e.g. (numbers in hex to make it easy to read)
- 12345678 can be stored in 4x8bit locations as follows

Byte Order (example)

Address	Value (1)	Value(2)
184	12	78
185	34	56
186	56	34
186	78	12

- i.e. read top down or bottom up?

Byte Order Names

- The problem is called Endian
- The system on the left has the least significant byte in the lowest address
- This is called big-endian
- The system on the right has the least significant byte in the highest address
- This is called little-endian

Example of C Data Structure

```
struct{
   int      a;     //0x1112_1314                    word
   int      pad;   //
   double   b;     //0x2122_2324_2526_2728          doubleword
   char*    c;     //0x3132_3334                    word
   char     d[7];  //'A','B','C','D','E','F','G'   byte array
   short    e;     //0x5152                         halfword
   int      f;     //0x6161_6364                    word
} s;
```

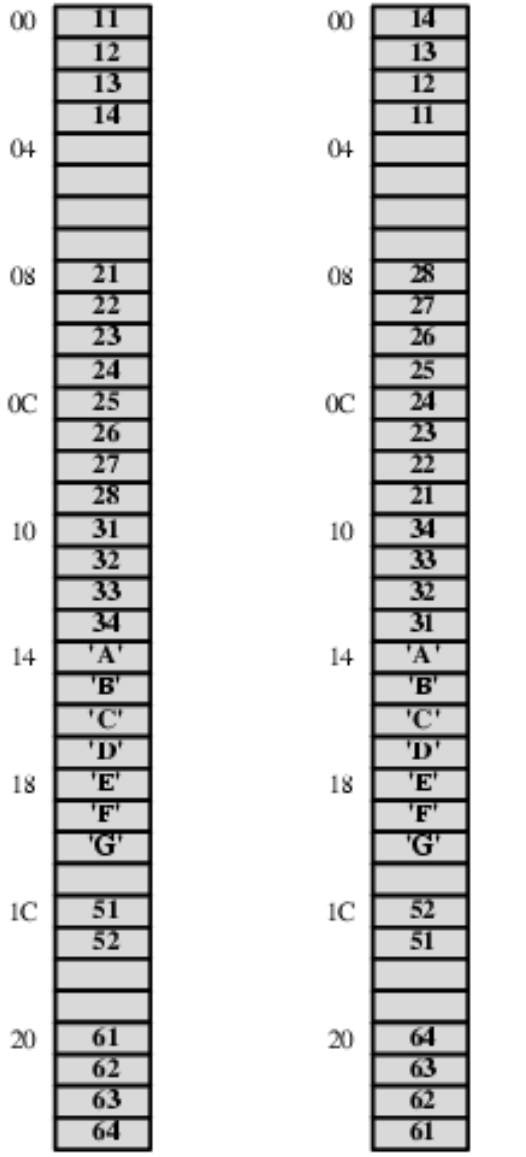
Big-endian address mapping

Byte Address								
00	**11** 00	**12** 01	**13** 02	**14** 03	04	05	06	07
08	**21** 08	**22** 09	**23** 0A	**24** 0B	**25** 0C	**26** 0D	**27** 0E	**28** 0F
10	**31** 10	**32** 11	**33** 12	**34** 13	**'A'** 14	**'B'** 15	**'C'** 16	**'D'** 17
18	**'E'** 18	**'F'** 19	**'G'** 1A	1B	**51** 1C	**52** 1D	1E	1F
20	**61** 20	**62** 21	**63** 22	**64** 23				

Little-endian address mapping

								Byte Address
00	07	06	05	04	**11** 03	**12** 02	**13** 01	**14** 00
08	**21** 0F	**22** 0E	**23** 0D	**24** 0C	**25** 0B	**26** 0A	**27** 09	**28** 08
10	**'D'** 17	**'C'** 16	**'B'** 15	**'A'** 14	**31** 13	**32** 12	**33** 11	**34** 10
18	1F	1E	**51** 1D	**52** 1C	1B	**'G'** 1A	**'F'** 19	**'E'** 18
20					**61** 23	**62** 22	**63** 21	**64** 20

Alternative View of Memory Map



(a) Big-endian (b) Little-endian

Standard…What Standard?

- Pentium (80x86), VAX are little-endian
- IBM 370, Moterola 680x0 (Mac), and most RISC are big-endian
- Internet is big-endian
  - —Makes writing Internet programs on PC more awkward!
  - —WinSock provides htoi and itoh (Host to Internet & Internet to Host) functions to convert