CSC 374: Computer Systems 2, 2010 Fall, Assignment #4

Last revised 2010 Oct 26

Purpose

To go over virtual memory/caching issues, paging, and safe C string/pointer usage.

Assignment

1. **Virtual Memory and the Cache (30 Points)**

   Please tell me the following about virtual address: ***0x02F9***

   - **Virtual page number:**
   - **Physical page number:**
   - **Page offset:**
   - **Cache Tag:**
   - **Cache Index:**
   - **Cache Offset:**
   - ***Byte at that position!***

   Assume, as in lecture 6:

   1. a 14-bit virtual address space,
   2. a 12-bit physical address space
   3. a page size of 64 bytes

```
Virtual Page:
13 12 11 10  9  8  7  6  5  4  3  2  1  0
|---------------------|  |--------------|
  Virtual page number    Virtual pg offset


Physical Page:
      11 10  9  8  7  6  5  4  3  2  1  0
      |---------------|  |--------------|
      Physical pg num    Physical pg offset
```

   The page table maps virtual pages to physical ones.
   Assume the following page table (all numbers in hexadecimal):

VPN	PPN	Valid?

00	28	1
01	--	0
02	33	1
03	02	1
04	--	0
05	16	1
06	--	0
07	--	0
08	13	1
09	17	1
0A	09	1
0B	23	1
0C	--	0
0D	2D	1
0E	11	1
0F	0D	1
. . .		

The Cache maps physical addresses to bytes stored there

```
Physical pg num    Physical pg offset
|---------------|  |--------------|
11 10  9  8  7  6  5  4  3  2  1  0
|---------------|  |--------|  |--|
       Tag            Index   Offset
```

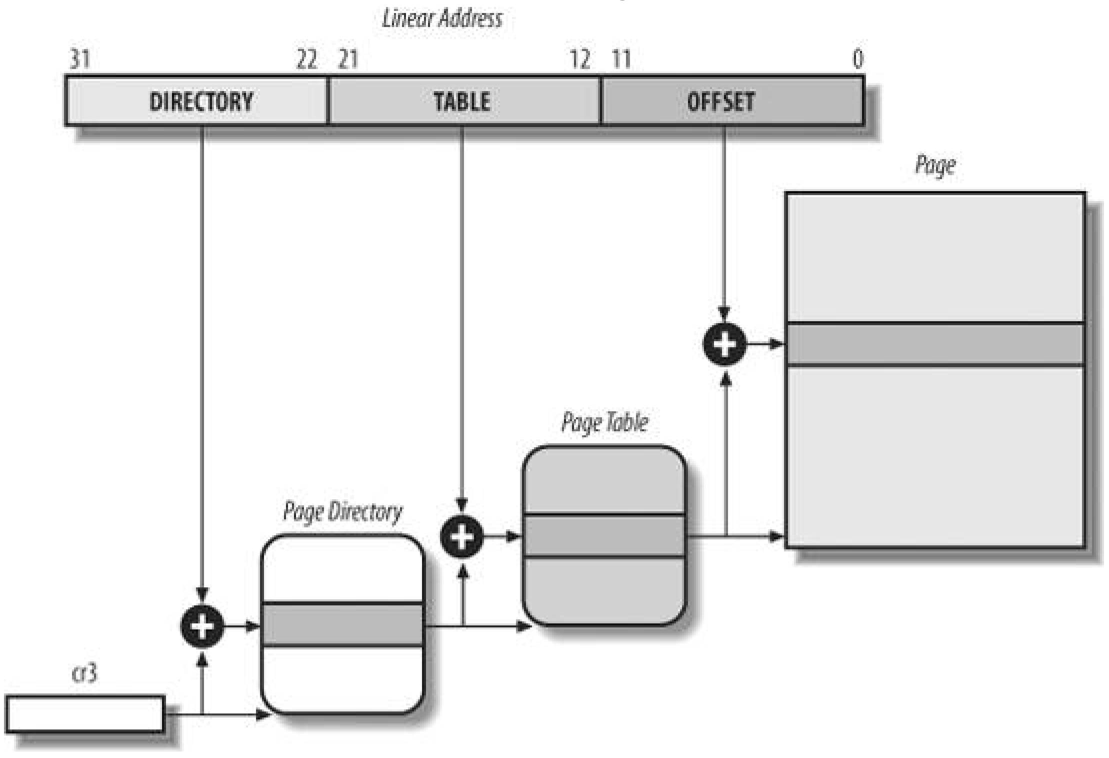
Index	Tag	Valid?	Byte offset			
			B0	B1	B2	B3
0	19	1	99	11	23	11
1	15	0	--	--	--	--
2	1B	1	00	02	04	08
3	36	0	--	--	--	--
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	--	--	--	--
7	16	1	11	C2	DF	03

8	24	1	3A	00	51	89
9	2D	0	--	--	--	--
A	2D	1	93	15	DA	3B
B	0B	1	41	32	23	14
C	12	0	--	--	--	--
D	16	1	04	96	34	15
E	23	1	83	77	1B	D3
F	14	0	--	--	--	--

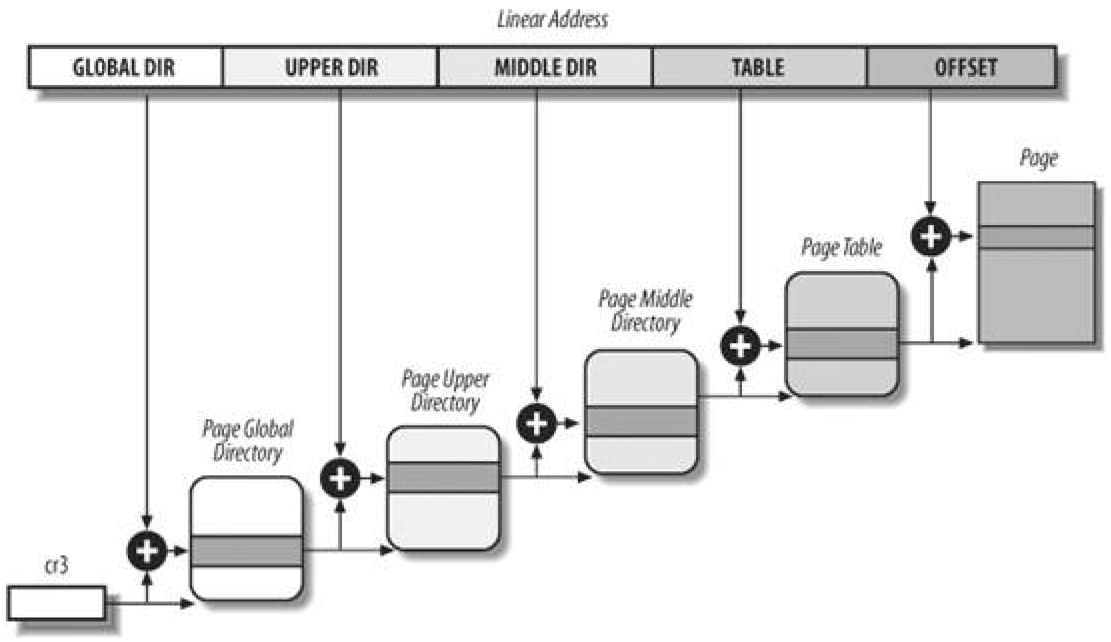
2. **64-Bit Paging (30 Points)**

   Recall that in going from a 32-bit address space to a 64-bit address space we doubled the number of small "page tables" from 2 to 4.

32 bits:



64
bits:



A. What was the reason we needed more small page tables?
   Why no still have two, but just make them larger?

B. If 4 is good, why not even more?
   Why not use 6 or 8 small tables that reference each other?

3. **Good C String Programming (40 Points)**

   Below there's a C program that is a **case study** in bad string programming in C. **Re-write it** into both:

   A. A *proper* C program that uses the C string library. *Be sure to use* `const` *pointers where appropriate!*
   B. A C++ program that uses the C++ string class.

```
/*-------------------------------------------------------------------------*
 *----                                                                 ----*
 *----          badString.c                                            ----*
 *----                                                                 ----*
 *----      This program serves as a case-study in how *not* to do     ----*
 *----  string programming in C.                                       ----*
 *----                                                                 ----*
 *----  ----    ----    ----    ----    ----    ----    ----    ----    ----*
 *----                                                                 ----*
 *----  Version 1.0             Joseph Phillips        2010 October 26 ----*
 *----                                                                 ----*
 *-------------------------------------------------------------------------*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

```
#define  STRING_LEN     20

#define  NUMBER_LEN     3



/*  PURPOSE:  To let the user enter their name into the array pointed to by
 *      'namePtr'.  No return value.
 */
void    enterName       (char*  namePtr)
{
  printf("Please enter your name: ");
  gets(namePtr);
}



/*  PURPOSE:  To let the user enter their age into the integer pointed to by
 *      'agePtr'.  No return value.
 */
void    enterAge        (int*   agePtr)
{
  char  numberText[NUMBER_LEN];

  printf("Please enter your age: ");
  gets(numberText);
  *agePtr = atoi(numberText);
}



/*  PURPOSE:  To let the user enter their favorite color for the item whose
 *      name is pointed to by 'itemNamePtr' into the space pointed to by
 *      'entryPtr'.  No return value.
 */
void    enterFavoriteColor      (char*  itemNamePtr,
                                 char*  entryPtr
                                )
{
  printf("Please enter your favorite color for a %s.  ",itemNamePtr);
  gets(entryPtr);
}



/*  PURPOSE:  To print out information on the user whose name is pointed to
 *      by 'namePtr', whose age is given in 'age', whose favorite car color's
 *      name is pointed to by 'carColorPtr' and whose favorite house color's
 *      name is pointed to by 'houseColorPtr'.  No return value.
 */
void    printInfo       (char*  namePtr,
                         int    age,
                         char*  carColorPtr,
                         char*  houseColorPtr
```

```
                        )
{
  char  designation[STRING_LEN];

  sprintf(designation,"%s who is %d years old",namePtr,age);
  printf ("%s likes the car color %s.\n",designation,carColorPtr);

  if  (strcmp(carColorPtr,houseColorPtr) == 0)
    printf("They do like the same color for houses, too.\n");
  else
    printf("However, they prefer houses colored %s.\n",houseColorPtr);

}



/*  PURPOSE:  To run the program.  Ignores parameters.  Returns 'EXIT_SUCCESS'
 *      to OS on completion.
 */
int     main ()
{
  char  name[STRING_LEN];
  int   age;
  char  carColor[STRING_LEN];
  char  houseColor[STRING_LEN];

  enterName(name);
  enterAge(&age);
  enterFavoriteColor("car",carColor);
  enterFavoriteColor("house",houseColor);
  printInfo(name,age,carColor,houseColor);
  return(EXIT_SUCCESS);
}
```

Not so bad output:

```
[jphillips@localhost Assign4]$ badString
Please enter your name: Joe
Please enter your age: 43
Please enter your favorite color for a car.  gray
Please enter your favorite color for a house.  brown
Joe who is 43 years old likes the car color gray.
However, they prefer houses colored brown.
```

Really bad output:

```
[jphillips@localhost Assign4]$ badString
Please enter your name: Joseph Perry Phillips
Please enter your age: 143
Please enter your favorite color for a car.  orange with shiny purple speckles
Segmentation fault
```