ארגון המחשב ומערכות הפעלה

אביב תשפ"ד

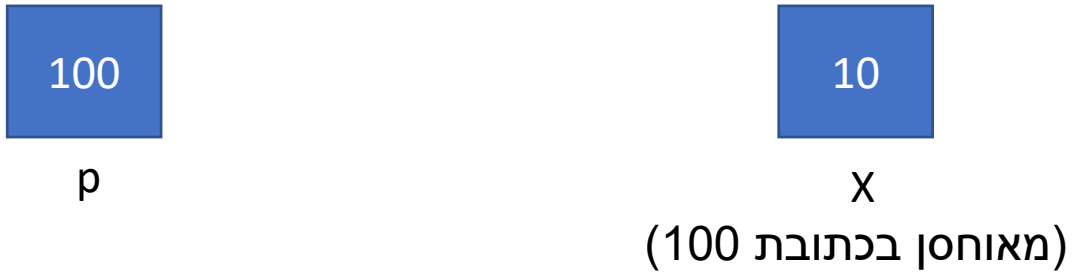
תרגול 3 – יישור קו ב C ומסכות

C

(יישור קו)

מצביעים

```
void show_pointers() {
  int* p; // p is a pointer
  int x = 10;
  int y;
  p = &x;  // p holds the address of x.
  printf("%d\n", *p); // 'dereferencing' p. prints "10"
  y = *p;
  x = 5;
  printf("%d %d\n", *p, y); // prints "5 10"
}
```



מבנים

```
struct point {
  int x;
  int y;
};

void show_struct() {
  struct point my_point;
  my_point.x = 0;
  my_point.y = 0;
}
```

הגדרת טיפוסים

```c
typedef int my_type;

void show_typedef() {
   my_type x = 3;
   printf("%d\n", x);
}
```

שילוב

```
typedef struct point{
  int x;
  int y;
};


void show_struct1() {
  point point;
  point *p;
  p = &point;
  p->x = 0; // same as (*p).x = 0;
  p->y = 0; // same as (*p).y = 0;
  printf("%d %d", p->x, p->y); // prints “0 0”
}
```

מסכות

בביתה למדנו על זיזות (shifting)

- זיזה שמאלה
- זיזה ימינה

אבל איך אפשר להשתמש בזיזות?

נגיד ואנחנו רוצים למצוא את הביט ה-6 במשתנה x מסוג int

$$x = \ldots\ldots\ldots \underbrace{0}_{6}\ \underbrace{1}_{5}\ \underbrace{1}_{4}\ \underbrace{0}_{3}\ \underbrace{1}_{2}\ \underbrace{0}_{1}\ \underbrace{1}_{0}$$

איך נוכל להגיד אם הוא 0 או 1?

מסכות

מסכה (mask)- וקטור בינארי שעוזר לחלץ מידע מווקטורים אחרים.

המסכה עוזרת לנו להגדיר על אילו ביטים אנחנו רוצים לשמור ואילו לא.

איך בונים מסכה?

- נשים 1-ים במקומות שנרצה לשמור
- נשים 0-ים במקומות שנרצה להוריד

פורמלית- ניקח את הספרה 1 ונבצע זיזה שמאלה עד שנגיע לביט הרצוי

נחזור לשאלה שלנו

נגיד ואנחנו רוצים למצוא את הביט ה-6 במשתנה x מסוג int

$$x = \ldots\ldots\ldots \overbrace{0}^{6}\overbrace{1}^{5}\overbrace{1}^{4}\overbrace{0}^{3}\overbrace{1}^{2}\overbrace{0}^{1}\overbrace{1}^{0}$$

נגדיר את המסכה mask להיות בגודל של המשתנה x

נרצה לשמור את הביט ה-6 ב-x לכן:

נשים 1 במקום ה-6 ב-mask וכל שאר המקומות יהיו אפסים

$$mask = 1 \ll 5$$

$$mask = \ldots\ldots\ldots \overbrace{0}^{6}\overbrace{1}^{5}\overbrace{0}^{4}\overbrace{0}^{3}\overbrace{0}^{2}\overbrace{0}^{1}\overbrace{0}^{0}$$

$$x = \overset{6}{\overbrace{0}}\ \overset{5}{\overbrace{1}}\ \overset{4}{\overbrace{1}}\ \overset{3}{\overbrace{0}}\ \overset{2}{\overbrace{1}}\ \overset{1}{\overbrace{0}}\ \overset{0}{\overbrace{1}}$$

$$mask = \overset{6}{\overbrace{0}}\ \overset{5}{\overbrace{1}}\ \overset{4}{\overbrace{0}}\ \overset{3}{\overbrace{0}}\ \overset{2}{\overbrace{0}}\ \overset{1}{\overbrace{0}}\ \overset{0}{\overbrace{0}}$$

נחזור לשאלה שלנו

עבשיו אנחנו רוצים למצוא את הביט ה-6 של x

בלומר:

- אם הביט ה-6 הוא 1 ← נרצה להחזיר 1
- אם הביט ה-6 הוא 0 ← נרצה להחזיר 0

איזו פעולה לוגית תיתן לנו את התוצאה הרצויה?

$$x = \overset{6}{0}\ \overset{5}{1}\ \overset{4}{1}\ \overset{3}{0}\ \overset{2}{1}\ \overset{1}{0}\ \overset{0}{1}$$

$$mask = \overset{6}{0}\ \overset{5}{1}\ \overset{4}{0}\ \overset{3}{0}\ \overset{2}{0}\ \overset{1}{0}\ \overset{0}{0}$$

נחזור לשאלה שלנו

נבצע פעולת & בין x ל mask. למה?

← אם הביט ה-6 ב-x הוא 1:

התוצאה תהיה וקטור שכולו 0-ים חוץ מהביט במקום ה-6 – התוצאה לא תהיה שקולה ל-0

← אם הביט ה-6 ב-x הוא 0:

התוצאה תהיה וקטור שכולו 0-ים

מכאן נוכל לכתוב את התוצאה res

$$res = (x\&mask)\ !=\ 0$$

תוציא true כאשר הביט ה-6 ב-x הוא 1 ותוציא false כאשר הביט שווה ל-0

מסכה מורכבת

- למדנו שאם אנחנו רוצים לגלות את הביט ה-n נבנה מסכה שתזיז שמאלה את הספרה 1 n-1 פעמים:

$$mask = 1U \ll (n-1)$$

- נגיד ואנחנו רוצים לשמור את n הביטים הראשונים במשתנה. נבנה מסכה מורכבת-בונים מסכה רגילה ומוסיפים לה 1-

$$mask = (1U \ll n) - 1$$

מוסיפים 1-

מזיזים שמאלה n פעמים

הספרה 1 ב-unsigned

זוכרים איך הווקטור 1- מיוצג? כולו 1-ים

דוגמה

בהנתן משתנה בגודל 32 סיביות (ביטים)- נרצה את 10 הביטים הראשונים

נבנה מסכה בגודל n=10

$$mask = (1U \ll 10) - 1$$

$$\mathbf{1U \ll 10}: \quad 000 \ldots \ldots 1000 \ldots \ldots 0$$

$$+$$

$$\mathbf{-1}: \quad 111 \ldots \ldots 1111 \ldots \ldots 1$$

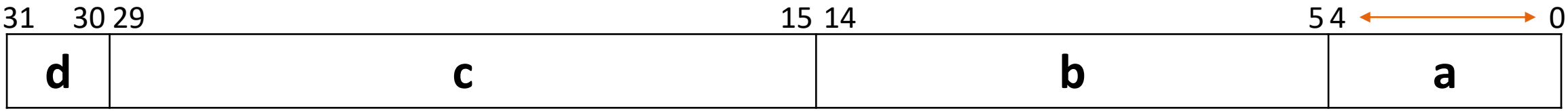
$$\underbrace{000 \ldots \ldots 0}\ \underbrace{111 \ldots \ldots 1}$$

10 הביטים הראשונים
שווים ל-1

קריאת ערכי משתנים בעזרת מסיכות

נניח משתנה unsigned בגודל 32 סיביות

הביטים שמייצגים את המשתנה



רוצים לקרוא את הערך b

מה השלבים?

1. נזיז את המשתנה ב-5 מקומות ימינה כדי שנתחיל מ-b
2. נבנה מסכה בגודל של הערך b (|b|=10)
3. נאפס את כל הערכים שאנחנו לא צריכים: c ו-d

המשך

1. נזיז את המשתנה ב-5 מקומות ימינה כדי שנתחיל מ-b

2. נבנה מסכה בגודל של הערך b (|b|=10): $mask = (1U \ll 10) - 1$

00..00	00	000.....000	11.....11

3. נאפס את כל הערכים שאנחנו לא צריכים (c ו-d) בעזרת הפעולה AND

00..00	00	000.....000	b

ועכשיו לפונקציה – קריאת ערך

```
unsigned get_field( int start, int length, unsigned x)
{
   unsigned res = x >> start ;
   unsigned mask = (1U << length) -1 ;
   res = res & mask ;
   return res ;
}
```

מזיזים את x ימינה start פעמים

בונים מסכה בגודל length

מורידים את החלקים שלא רוצים

Start - הביט בו השדה b מתחיל
Length - האורך של השדה b
x - הוקטור השלם

כתיבה למשתנים בעזרת מסיכות

עכשיו רוצים גם לכתוב במקום של b את הערך val

1. נבנה <u>מסכה מאפסת</u>:
   הפוך ממסכה רגילה. 0-ים במקומות שנרצה לאפס ו1-ים ביתר המקומות.
2. נאפס את הערך b באמצעות פעולת AND
3. נזיז את הערך val למקום של b
4. נבצע את פעולת OR כדי לקבל את val בווקטור x במקום b

איך בונים מסכה מאפסת? בונים מסכה רגילה ועושים NOT על התוצאה

1. נבנה <u>מסכה מאפסת</u>:

הפוך ממסכה רגילה. 0-ים במקומות שנרצה לאפס ו1-ים ביתר המקומות.

11	**111.....111**	**000.....000**	**11.....11**

mask

2. נאפס את הערך b

d	**c**	**000.....000**	**a**

x

3. נזיז את הערך val למקום של b

00	**000.....000**	**val**	**00.....00**

val

4. נבצע את פעולת OR כדי לקבל את val בוקטור x במקום b

d	**c**	**val**	**a**

x

ועכשיו לפונקציה – כתיבת ערך

```
void set_field( int start, int length, unsigned x, unsigned val)
{
   unsigned mask = (1U << length) -1 ;
   mask = mask << start ;
   x = (x & ~mask) ;
   val = val << start ;
   x = (x | val) ;
}
```

בונים מסכה בגודל length

מזיזים start פעמים שמאלה למיקום של b

~mask : המסכה המאפסת- מאפסת את הביטים ב-b

מזיזים את start val פעמים שמאלה למיקום של הביטים של b

מכניסים את val במיקום של b לוקטור x

תרגיל 1

כתוב ביטוי ב-C שמקבל שני ערכים x,y ומוציא ביטוי שמורכב
מה-least significant byte (LSB) של x וכל שאר הביטים מ-y

דוגמה: עבור

$$x = 0x89ABCDEF$$
$$y = 0x76543210$$

התוצאה תהיה $0x765432EF$

תרגיל 1 - פתרון

- נבנה מסכה בגודל 8- נחלץ את ה LSB של x ונאפס את ה LSB של y
- התוצאה תהיה x|y

```
mask= (1<<8)-1;
x = x&mask;
y = y&~mask;
res= x|y;
```

חילוץ

איפוס

תרגיל 2

לכל אחד מ4 הסעיפים שלפניכם, כתבו ביטוי ב-C שמחזיר:

- 1 אם התנאי מוציא true
- 0 אם התנאי מוציא false

נניח שהמשתנה x הוא מסוג int.

אי אפשר להשתמש ב- == או =!

1. לפחות אחד מהביטים ב-x הוא 1
2. לפחות אחד מהביטים ב-x הוא 0
3. לפחות אחד מהביטים ב-LSB של x הוא 1
4. לפחות אחד מהביטים ב-MSB של x הוא 0

תרגיל 2 - פתרון

1. !!x
2. !!(~x)
3. A. !!(x&mask) mask = (1U <<8) -1
   B. !!(x & 0xFF) 0xFF = 11111111
4. !!(~(x>>24)) בהנחה שאנו בייצוג של 32 סיביות