

ייצוג משתתפים פרימיטיביים

וקבצים בינריים

# סיביות וכתמים

• כל המידע במחשב מיוצג בסיביות בינריות ( binary digits, bits)

• הסיביות ארוזות בכל המחשבים המודרניים בקבוצות של 8 סיביות שנקראות בתים (bytes)

$$00100011_2 = 32_{10} + 2_{10} + 1_{10} = 35_{10}$$

• בתוכניות ותייעוד, גם בסיס 16 נפוץ; המספרים נקראים הקסה-דצימליים, או הקסה (hexadecimal, hex); בג'אווה, C, ושפות אחרות מסמנים קבועים בבסיס 16 בקידומת 0x,

$$00100011_2 = 23_{16} = 0x23 = 35_{10}$$

• ב-8 סיביות אפשר לייצג את השלמים 0 עד 255,

$$11111111_2 = 128_{10} + 64_{10} + \dots = 255_{10} = 0xFF$$

## שלמים שליליים במשלים 2

• במקום השלמים 0 עד 255, ניתן לייצג בבית אחד את השלמים  
-128 עד 127

• הייצוג הזה נקרא משלים 2

• כאשר הסיבית העליונה דלוקה, המספר שלילי

10011101 → negative

• המספר השלילי שמיוצג על ידי בית הוא הנגדי למספר שצריך  
להוסיף למספר החיובי המיוצג כדי להגיע ל-256

$$10011101_2 = 128+16+8+4+1 = 157_{10}$$

$$10011101 = 0xAD = -(256 - 157) = -99_{10}$$

$$0xFF = -(256 - 255) = -1_{10}$$

## ייצוג שלמים ב-2, 4, ו-8 בתים

- אם צריך לייצג שלמים בטווח גדול יותר מ-0 עד 255 או מ-128 עד 127, משתמשים ביותר בתים

- בג'אווה: short הוא בית אחד, short שניים, int ארבעה, long שמונה (כולם עם סימן)

- בג'אווה char הוא תו יוניקוד אבל גם מספר שלם בלי סימן שמיוצג על ידי 2 סיביות (0 עד 65536)

- דוגמה של short:

$$01000001\ 10000011 = 0x4183 =$$

$$4x4096 + 1x256 + 8x16 + 3x1 = 16771_{10}$$

- בכל פורמט כזה עם סימן מספרים שליליים מיוצגים על ידי סיבית עליונה דלוקה

# תוים, Unicode, ASCII

- תווים (אותיות וסימנים) מיוצגים במחשב על ידי מספרים
- סט התווים שניתן לייצג נקרא `character set`
- המיפוי ממספרים, בדרך כלל שלמים חיוביים, לתווים, נקרא קידוד, `coding`
- סט התווים של ג'אווה והקידוד שלהם מבוסס על סטנדרט `Unicode` בשם
- תסבוכת קלה (בד"כ לא רלוונטית): ב-`Unicode` יש תווים עם קידוד גדול מ-65536, אבל בג'אווה `char` הוא שני בתים; לפעמים צריך לייצג תו על ידי שני `chars` רצופים במחרוזת
- `ASCII`: קידוד של סט תווים קטן (מספרים, סימני פיסוק בסיסיים, ואותיות לטיניות) ב-7 סיביות; תואם `Unicode` (כלומר תואם לאותיות `Unicode` עם קידוד קטן מ-128)

# קבצים בינריים וערוצי תקשורת

- קובץ מורכב מבתים
- לבתים יש סדר (ראשון, שני, ... , אחרון)
- גם כאשר מידע מגיע ממחשב אחר או תוכנית אחרת דרך ערוץ תקשורת, המידע מגיע כרצף סדור של בתים
- כיצד ממפים את רצף הבתים למבני נתונים מורכבים?
- קבצי טקסט מכילים מחרוזת אחת, אולי מופרדת לשורות
- קבצים בינריים מכילים מידע מסוגים שונים: שלמים אי שליליים ושלמים עם סימן, באורכים שונים, מחרוזות, מצביעים, ועוד
- קבצים מסוגים שונים שמורים כך: doc, class, zip, jpeg, ...

# מידע בקבצים בינריים

- שלמים של בית בודד עם סימן: הערך של ה-byte בג'אווה
- שלמים של בית בודד אי שלילי: ה-byte בג'אווה יכיל מספרים שליליים כאשר המספר שמיוצג בקובץ גדול מ-127
- בתוכנית צריך לייצג גדלים כאלה כ-short או int, כי byte לא מסוגל לייצג מספרים גדולים מ-127
- שלמים של 2, 4, או 8 בתים יכולים להיות שמורים בשתי צורות

- big endian (נקרא לפעמים Motorola order): הבית שמכיל את הסיביות היותר משמעותיות מופיע ראשון, למשל  $16771 = 0x4183$  יהיה מיוצג על ידי הרצף (משמאל לימין)  
 $0x41 \rightarrow 0x83 \rightarrow \dots$

# מידע בקבצים בינריים (המשך)

- `little endian` (Intel order): הפוך, 8 הסיביות הפחות משמעותיות שמורות ראשונות:  $16771 = 0x4183$  מיוצג על ידי הרצף

$0x83 \rightarrow 0x41 \rightarrow \dots$

- כאשר יש צורך בהצבעה לחלק של מבנה הנתונים שאינו מופיע מייד בהמשך הרצף, משתמשים במצביע: מספר שלם שמייצג את המרחק של החלק המוצבע של מבנה הנתונים מנקודה נתונה בקובץ (לא תמיד ההתחלה); בדרך כלל המרחק נמדד בבתים; אם המרחק עשוי להיות גדול, המצביע צריך להיות מיוצג ביותר מבית בודד
- יש מגוון גדול של ייצוג למחרוזות
- ייצוג פשוט הוא רצף `ASCII`, עם אורך או אפס מסיים