COURSE : FUNDAMENTALS OF COMPUTER SCIENCE

LESSON 3: "DATA STORAGE AND OPERATIONS ON DATA"

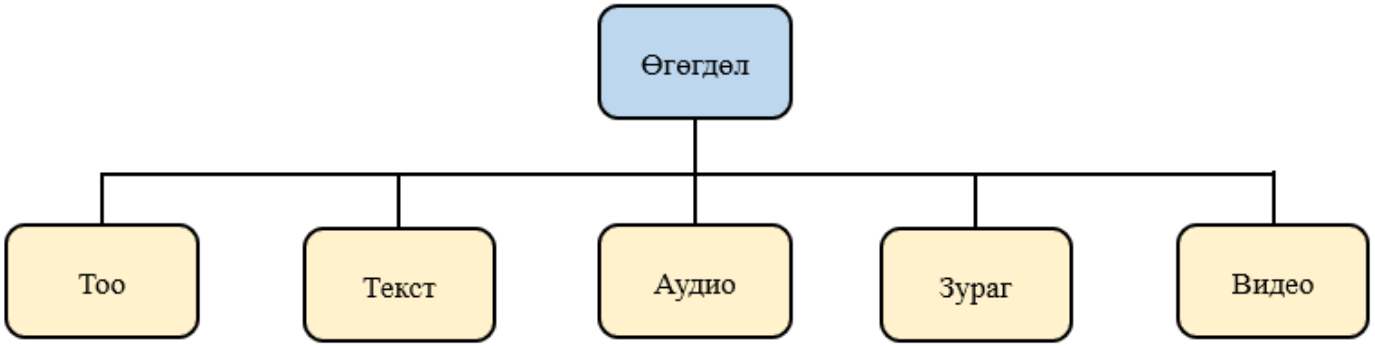
Instructor:
PhD, Associate Professor
Tuyatsetseg Badarch
SICT, MUST

Оршил

- Компьютер дотор өөр өөр өгөгдлийг хэрхэн хадгалагдаж байгааг тайлбарлах.
- Компьютерт бүхэл тоонууд /integer/ хэрхэн хадгалагдаж байгааг тайлбарлах.
- Төрөл бүрийн кодчилолын системийн аль нэгийг ашиглан компьютерт текст хэрхэн хадгалагдаж байгааг тайлбарлах
- Растер ба вектор графикийн схемийг ашиглан дүрсийг компьютерт хэрхэн хадгалдаг талаар..
- Видеог компьютерт хэрхэн хадгалж байгааг, цаг хугацааны хувьд өөрчлөгдөж буй зургуудын дүрслэл болгодог тухай тайлбарлах.
- Компьютерийн дотоод өгөгдлийн тооцооллын талаар судлах

Ангилал

Өгөгдөл нь тоо, текст, аудио, зураг, видео гэх мэт янз бүрийн хэлбэртэй байдаг (зураг 3.1).



Зураг 3.1 Ялгаатай өгөгдлийн төрлүүд

Компьютерийн салбар нь тоо, текст, зураг, аудио, видео агуулсан мэдээллийг тодорхойлохдоо "мультимедиа" гэсэн нэр томъёог ашигладаг.

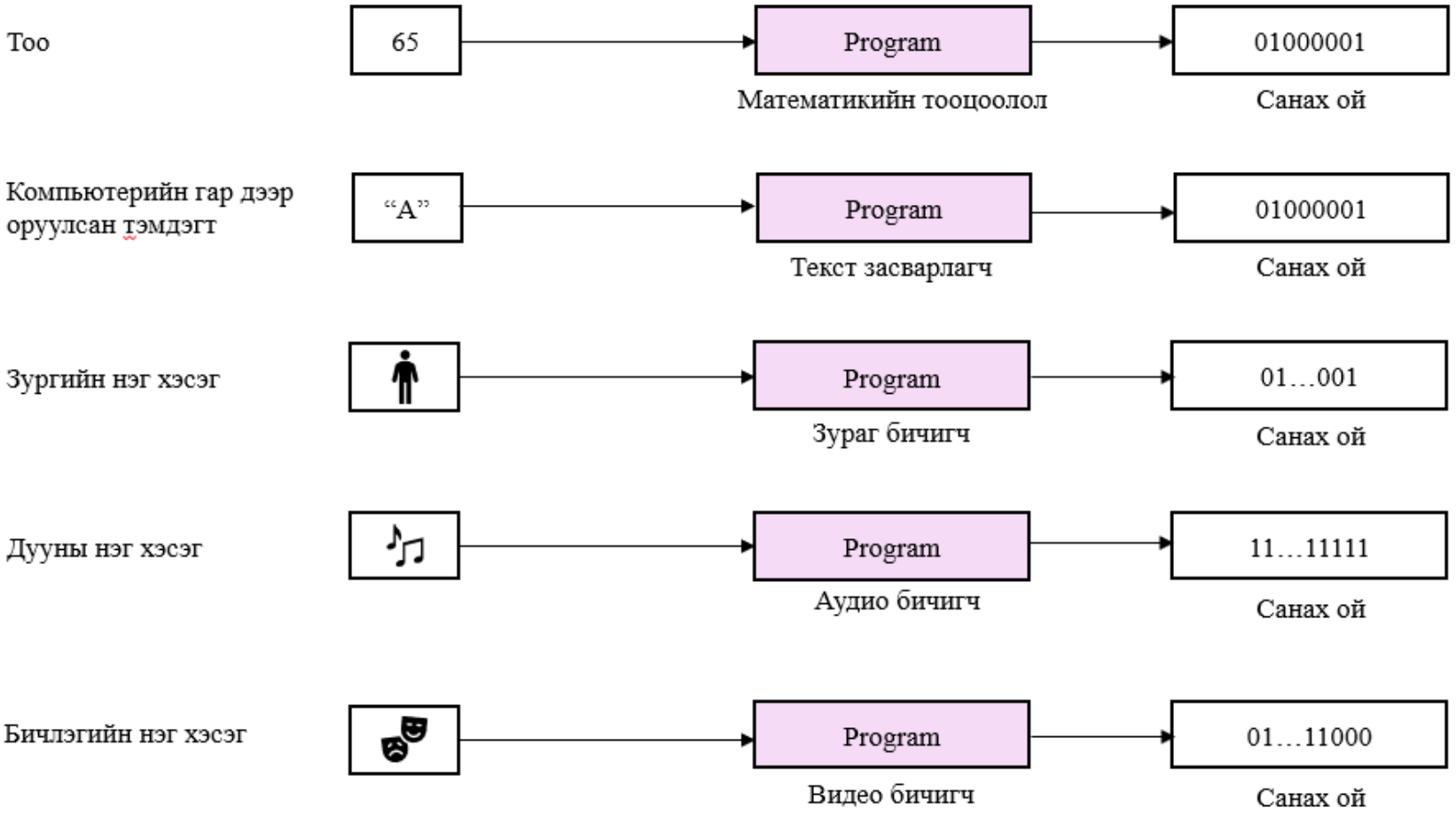
Компьютер дах өгөгдөл

Бүх төрлийн өгөгдлийг компьютерт хадгалж, буцааж хувиргаж анхны хэлбэрт шилжүүлснээр нэг төрлийн дүрслэл болгон хувиргадаг. Энэхүү ерөнхий дүрслэлийг бит загвар /bit pattern/ гэж нэрлэдэг. (битүүдийн дараалал – string of bits)

1 0 0 0 1 0 1 0 1 1 1 1 1 1

Зураг 3.2 16 бит бүхий загвар /bit pattern/

Компьютер дахь өгөгдөлийн хадгалалт



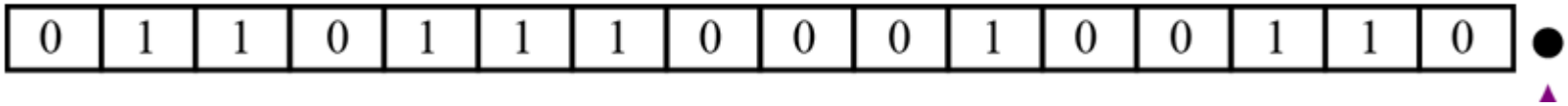
Зураг 3.3 Төрөл бүрийн өгөгдлийн хадгалалт

Компьютер дахь өгөгдөлийн хадгалалт

Аливаа тоог компьютерийн санах ойд хадгалагдахаас өмнө хоёртын систем руу хөрвүүлдэг боловч шийдвэрлэх шаардлагатай хоёр асуудал байсаар байна. Үүнд:

1. Тооны тэмдгийг /эерэг ба сөрөг/ хэрхэн хадгалах вэ?
2. Аравтын бутархайг хэрхэн дүрслэн харуулах вэ?

Компьютер дахь өгөгдөлийн хадгалалт



Зураг 3.4 Бүхэл тоонуудын цэгэн дүрслэл

Бүхэл тоо нь бүхэл тоо бөгөөд (бутархай хэсэггүй тоо). Жишээлбэл, 134 ба −125 нь бүхэл тоо, харин 134.235 ба −0.235 нь биш юм.

Аравтын бутархайн таслалаас өмнө байрласан тоог бүхэл тоо гэж ойлгож болно.

Үүний улмаас /Зураг 3.4-т үзүүлсэн/ бүхэл тоог хадгалахын тулд тогтмол цэгийн дүрслэлийг ашигладаг. Энэ дүрслэлд аравтын бутархайг тооцсон боловч хадгалаагүй болно.

Бүхэл тоог ерөнхий тохиолдолд тогтмол цэгийн дүрслэл ашиглан санах ойд хадгалагддаг.

Компьютер дахь өгөгдөлийн хадгалалт

Тэмдэггүй дүрслэл /unsigned integer/

Тэмдэггүй бүхэл тоо нь хэзээ ч сөрөг байж болохгүй бөгөөд зөвхөн 0, эерэг утгыг авах боломжтой бүхэл тоо юм. Түүний хүрээ нь 0 ба эерэг хязгааргүй хооронд байна.

Оролтын төхөөрөмж нь **unsigned integer** тоог дараах алхмуудыг ашиглан хадгална.

1. Бүхэл тоог хоёртын тоо болгон өөрчлөнө.
2. Хэрэв битийн тоо n-ээс бага бол зүүн талд 0-ийг нэмнэ.

Компьютер дахь өгөгдөлийн хадгалалт

Жишээ 3.1

7-г unsigned int төрлөөр 8-битийн санах ойд хадгалах .

Бодолт

Эхлээд бүхэл тоог хоёртын хэлбэрт шилжүүл (111)

Таван 0-ийг нэмж нийт найман битийг (00000111) үүсгэнэ.

Компьютер дахь өгөгдөлийн хадгалалт

258-г 16 битийн санах ойд хадгал.

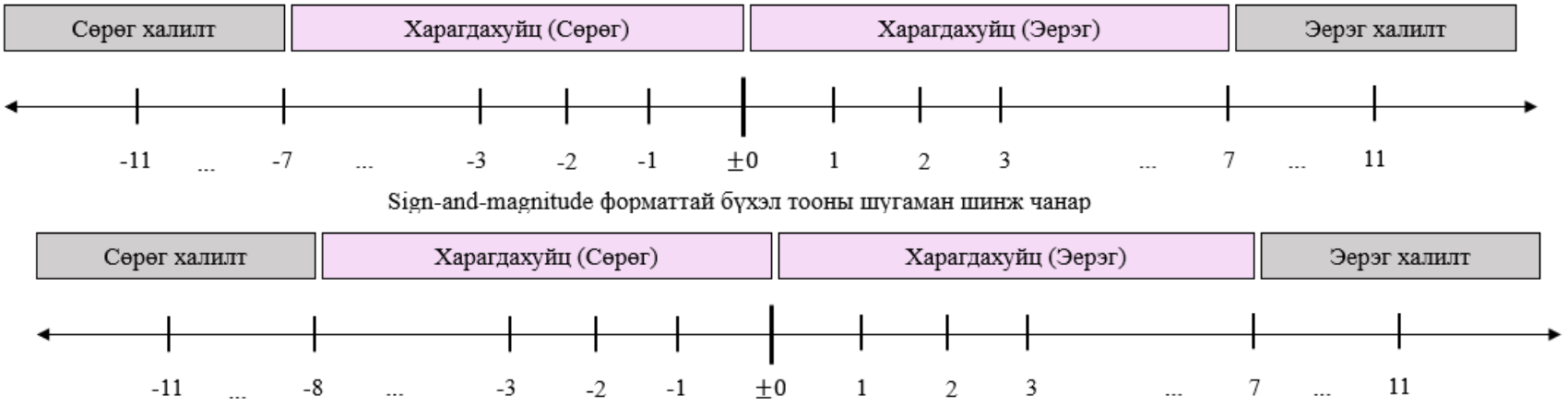
Бодолт

Эхлээд бүхэл тоог хоёртын (100000010) тоо болгож өөрчлөнө. Долоон 0-ийг нэмээд нийт арван зургаан бит , (0000000100000010) Бүхэл тоо санах ойн байршилд хадгалагдана.

							1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0

Компьютер дахь өгөгдөлийн хадгалалт

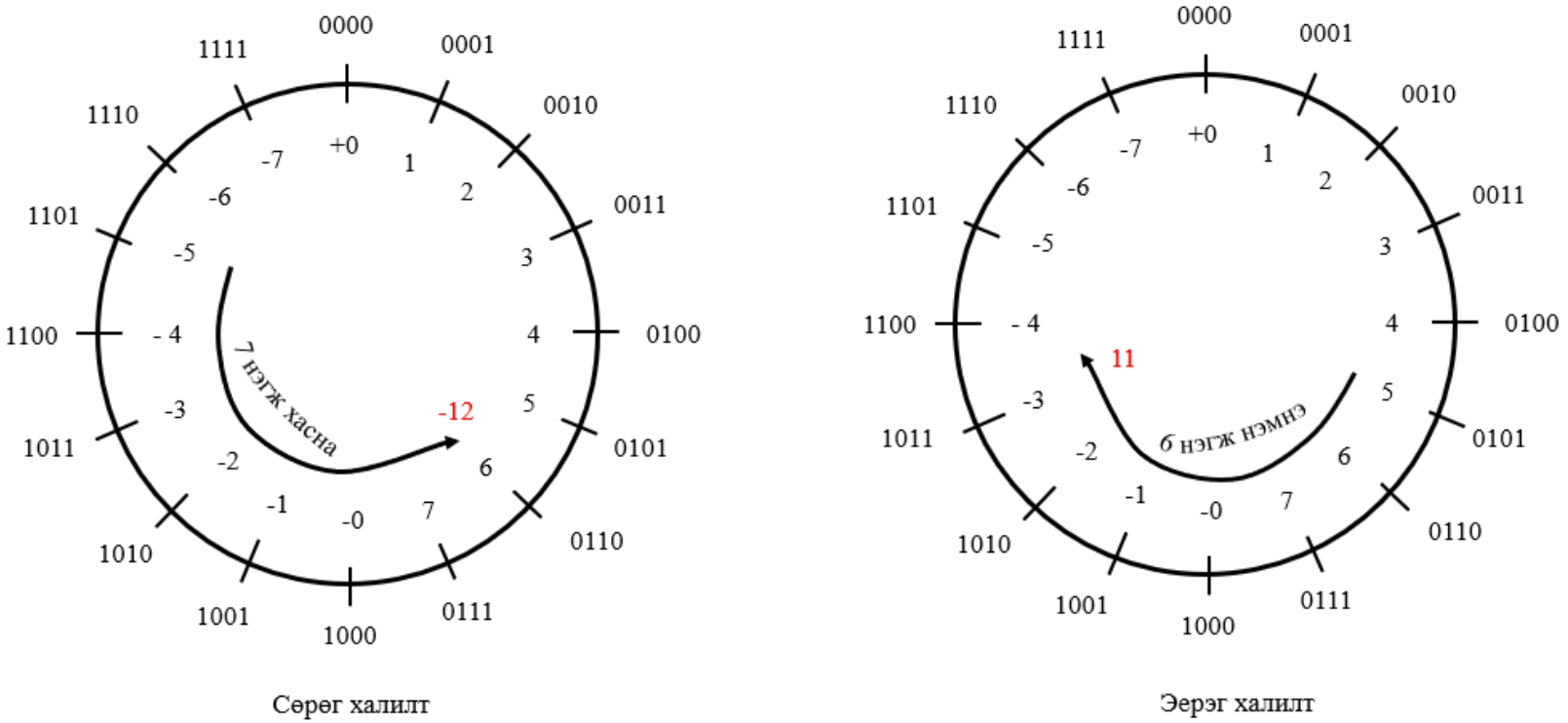
Хэрэв бид зөвхөн 4 битийн санах ойд $2^4 - 1 = 15$-оос их тоо бүхэл тоо /integer/-г хадгалах тохиолдлыг Зураг 3.5а-д үзүүллээ.



Зураг 3.5а **Тэмдэггүй бүхэл тоон орон халилт**

Компьютер дахь өгөгдөлийн хадгалалт

Хэрэв бид зөвхөн 4 битийн санах ойд $2^4 - 1 = 15$-оос их тоо бүхэл тоо /integer/-г хадгалах тохиолдлыг Зураг 3.5b-д үзүүллээ.



Зураг 3.5b **Тэмдэггүй бүхэл тоон орон халилт**

Компьютер дахь өгөгдөлийн хадгалалт

-28 тоог / sign-and-magnitude/-аар дүрсэлж 8 битийн санах ойд хадгал.

Бодолт

Бүхэл тоо нь 7 битийн хоёртын хэлбэртэй болох ба хамгийн зүүн талын битийг “1” гэж тохируулан 8-бит тоогоор хадгалагдана.

Change 28 to 7-bit binary		0	0	1	1	1	0	0
Add the sign and store	1	0	0	1	1	1	0	0

Компьютер дахь өгөгдлийн хадгалалт

Жишээ 3.7

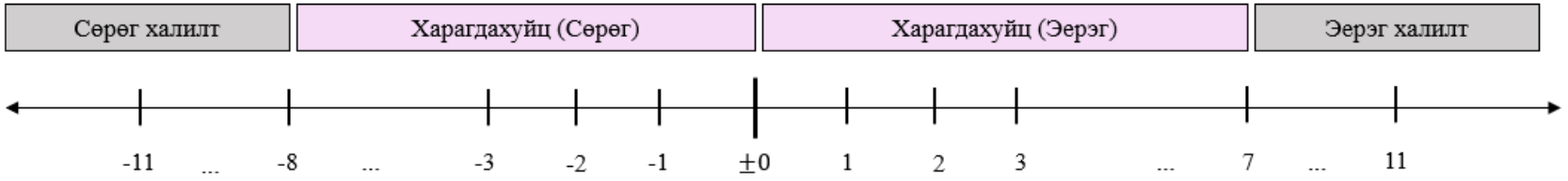
“10100001” гэж хадгалагдсан бүхэл тоог буцаан /sign-and-magnitude/ -аар дүрсэл

Бодолт:

Хамгийн зүүн талын бит 1 тул тэмдэг сөрөг байна. Үлдсэн битүүдийг (0100001) аравтын тоо руу хөрвүүлбэл 33 болно. Тэмдгийг нэмсний дараа бүхэл тоо −33 болно.

Компьютер дахь өгөгдөлийн хадгалалт

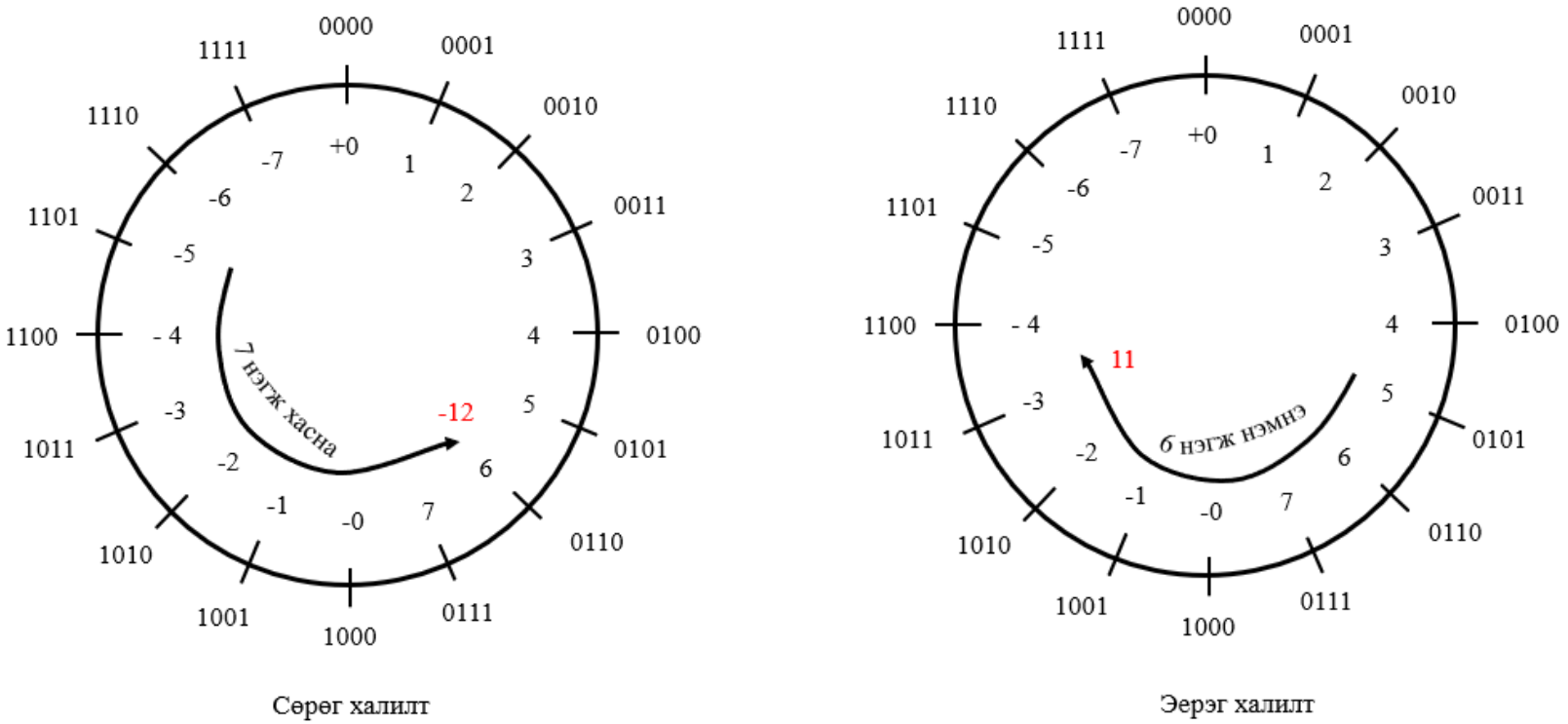
Зураг 3.7а-д 4 битийн санах ойд эерэг ба сөрөг бүхэл тоог /sign-and-magnitude /-аар хадгалах үеийн overflow-ийг дүрслэв.



Зураг 3.7а sign-and-magnitude –аар илэрхийлэх Overflow

Компьютер дахь өгөгдөлийн хадгалалт

Зураг 3.7b-д 4 битийн санах ойд эерэг ба сөрөг бүхэл тоог /sign-and-magnitude /-аар хадгалах үеийн overflow-ийг дүрслэв.



Зураг 3.7b sign-and-magnitude –аар илэрхийлэх Overflow

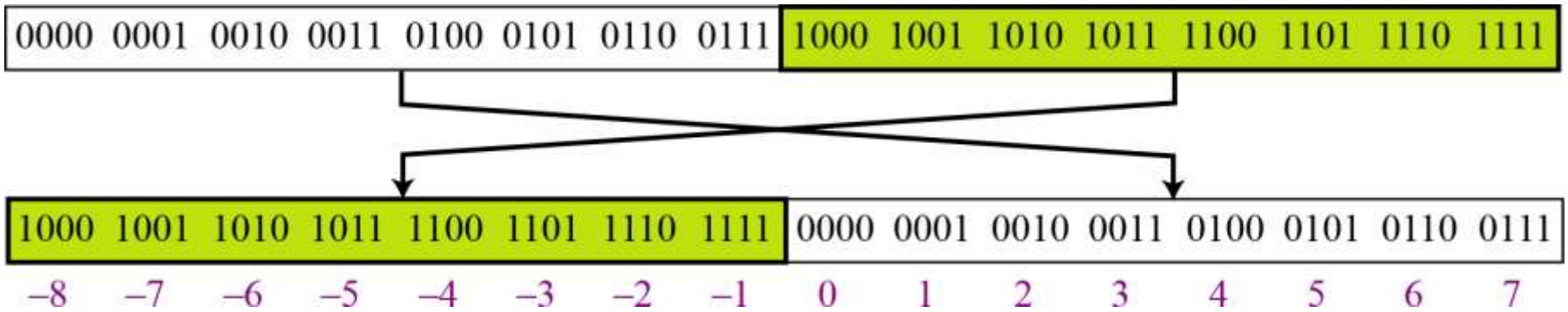
2-н гүйцээлтийн дүрслэл

- Бараг бүх компьютерийн n-bit санах ойд тэмдэгт бүхэл тоог хадгалахын тулд хоёрын гүйцээлт дүрслэлийг ашигладаг.

- Энэ аргын хувьд (0-ээс 2n - 1) гэсэн тэмдэггүй бүхэл тооны боломжтой мужийг хоёр тэнцүү дэд мужид хуваана. Эхний дэд муж нь сөрөг бус бүхэл тоонуудыг, хоёр дахь хагасыг сөрөг бүхэл тоог илэрхийлэхэд ашигладаг.

- Дараа нь битийн хэв маягийг Зураг 3.8-д үзүүлсний дагуу сөрөг ба сөрөг бус (тэг ба эерэг) бүхэл тоонуудад хуваарилна.

Компьютер дахь өгөгдөлийн хадгалалт



Зураг 3.8 2-ын гүйцээлтийн дүрслэл

Хоёрын гүйцээлтийн дүрслэлд хамгийн зүүн талын бит нь бүхэл тоон тэмдгийг тодорхойлно. Хэрэв 0 бол бүхэл тоо эерэг байна. Хэрэв энэ нь 1 бол бүхэл тоо сөрөг байна.

Компьютер дахь өгөгдөлийн хадгалалт

бүхэл тооны нэгийн гүйцээлт үйлдлийг эерэг эсвэл сөрөг аль ч бүхэл тоонд хэрэглэж болно. Энэ үйлдэл нь ердөө л бит тус бүрийг инверслэнэ (эргүүлнэ). 0-битийг 1-бит, 1-битийг 0-бит болгон өөрчилнө.

Нэгийн гүйцээлт

00110110 бүхэл тооны нэгийн гүйцээлтийг дараах байдлаар харуулав.

Original pattern	0	0	1	1	0	1	1	0
After applying one's complement operation	1	1	0	0	1	0	0	1

Компьютер дахь өгөгдөлийн хадгалалт

Хэрвээ анхны бүхэл тооноос хоёр удаа 1-ийн гүйцээлт авбал доорх маягаар дүрслэнэ

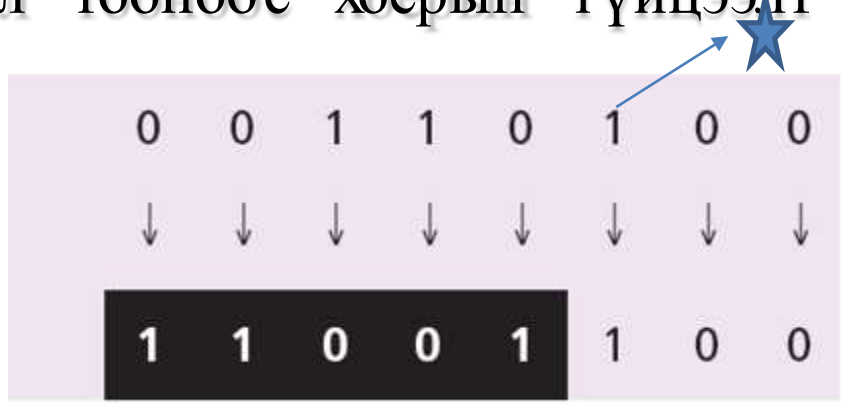
Компьютер дахь өгөгдөлийн хадгалалт

2-ын гүйцээлт

Хоёр дах үйлдлийг нь ***two's complementing*** буюу 2-тын бүхэл тоон 2-ын гүйцээлт авах гэж нэрлэдэг. Энэ үйлдэл нь хоёр алхамаар гүйцэтгэгдэнэ. 1-рт, хамгийн баруун талын "1" утгатай хүртэлх битүүдийг инверс хийнэ. Үлдсэн битүүдийг хэвээр

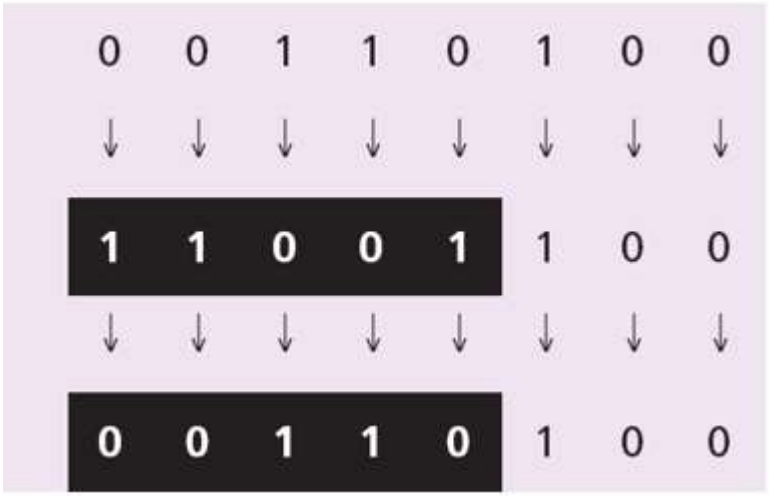
Жишээ 3.10

00110100 бүхэл тооноос хоёрын гүйцээлт хэрхэн авахыг харуулав.



Компьютер дахь өгөгдөлийн хадгалалт

Хэрэв бид хоёрын гүйцээлтийг хоёр удаа хийвэл бид үргэлж анхны бүхэл тоог гарган авах болохыг дараах байдлаар харуулж байна.



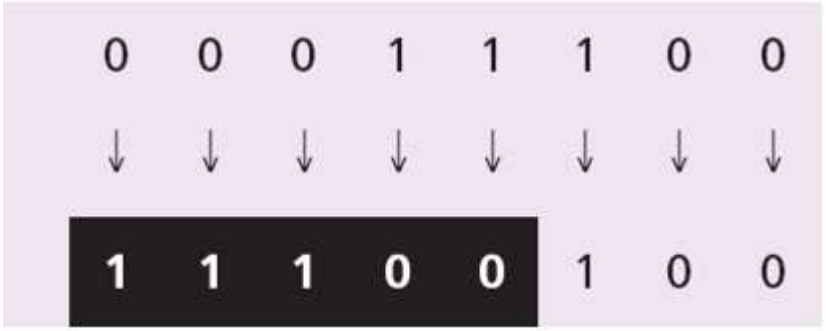
Бүхэл тооны хоёрын гүйцээлт авах өөр арга бол эхлээд нэгийн гүйцээлтийг аваад үр дүн дээр нь +1 нэмэх явдал юм.

Компьютер дахь өгөгдлийн хадгалалт

-28 гэсэн бүхэл тоог хоёрын гүйцээлтийн илэрхийлэл ашиглан 8 бит санах ойд хадгал.

Бодолт

Бүхэл тоо нь сөрөг тул хоёртын тоо руу хөрвүүлсний дараа бүхэл тоонд хоёрын гүйцээлт хийнэ



Компьютер дахь өгөгдлийн хадгалалт

Scientific notation буюу ШУ-ны тэмдэглэгээ

Жишээ 3.18

Доорхи аравтын тоог үзүүлэв.

7,452,000,000,000,000,000,000.00

шинжлэх ухааны тэмдэглэгээнд (хөвөгч таслалын дүрслэл) scientific notation (floating-point representation):.

Actual number	→	+	7,425,000,000,000,000,000,000.00
Scientific notation	→	+	7.425×10^{21}

Гурван хэсэг нь тэмдэг (+), шилжүүлэгч (21) ба тогтмол цэгийн хэсэг (7.425) юм. Шилжүүлэгч нь экспонент болохыг анхаарна уу.

Компьютер дахь өгөгдлийн хадгалалт

Жишээ 3.19 **–0.0000000000000232** шинжлэх ухааны тэмдэглэгээнд (хөвөгч таслалын дүрслэл).

Шийдэл

Бид өмнөх жишээн дээр дурдсантай ижил аргыг ашиглана - аравтын бутархайн таслалаас хойш 2 цифр байхаар авна.

Actual number	→	–	0.0000000000000232
Scientific notation	→	–	2.32×10^{-14}

Гурван хэсэг нь тэмдэг (-), шилжүүлэгч (-14) ба тогтмол цэгийн хэсэг (2.32) байна. Шилжүүлэгч нь экспонент болохыг анхаарна уу.

Компьютер дахь өгөгдлийн хадгалалт

Тоог хөвөгч таслалтай илэрхийллээр үзүүл.

Жишээ 3.20

$$(10100100000000000000000000000000.00)_2$$

Бодолт

Бид аравтын бутархайн таслалын зүүн талд зөвхөн нэг оронтой тоог хадгалсантай адил бодож болно.

Actual number	→	+	$(10100100000000000000000000000000.00)_2$
Scientific notation	→	+	1.01001×2^{32}

Компьютер дахь өгөгдлийн хадгалалт

Тоог хөвөгч таслалтай илэрхийллээр үзүүл.

$-(0.000000000000000000000101)_2$

Бодолт

Бид аравтын бутархайн таслалын зүүн талд зөвхөн нэг оронтой тоог хадгалсантай адил бодож болно.

.

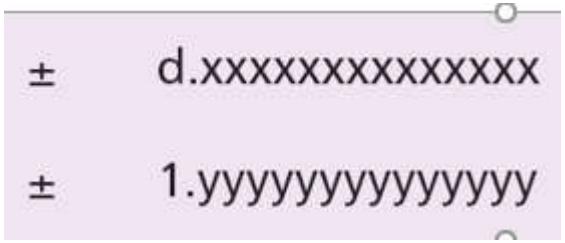
Actual number	→	–	$(0.000000000000000000000101)_2$
Scientific notation	→	–	1.01×2^{-24}

Компьютер дахь өгөгдлийн хадгалалт

Normalization

Илэрхийллийн тогтсон хэсгийг жигд болгохын тулд шинжлэх ухааны арга (аравтын тооллын системийн хувьд), хөвөгч таслалын арга (хоёртын системийн хувьд) хоёулаа аравтын бутархай таслалын зүүн талд тэгээс бусад цифрийг л ашигладаг. Үүнийг **normalization** гэж нэрлэдэг.

Аравтын системд 1-ээс 9 хүртэл цифр байж болох бол хоёртын системд зөвхөн 1 байж болно. Дараахь зүйлд d нь тэг биш, x нь цифр, y нь 0 эсвэл 1-ийн аль нэг нь байна.



Компьютер дахь өгөгдлийн хадгалалт

Sign, exponent and mantissa Тэмдэг, илтгэгч ба мантисс

+	2^6	×	1.0001110101
+	6		0001110101
↑	↑		↑
Sign	Exponent		Mantissa

Компьютер дахь өгөгдөлийн хадгалалт

Илүүдэл тооцох систем Excess System

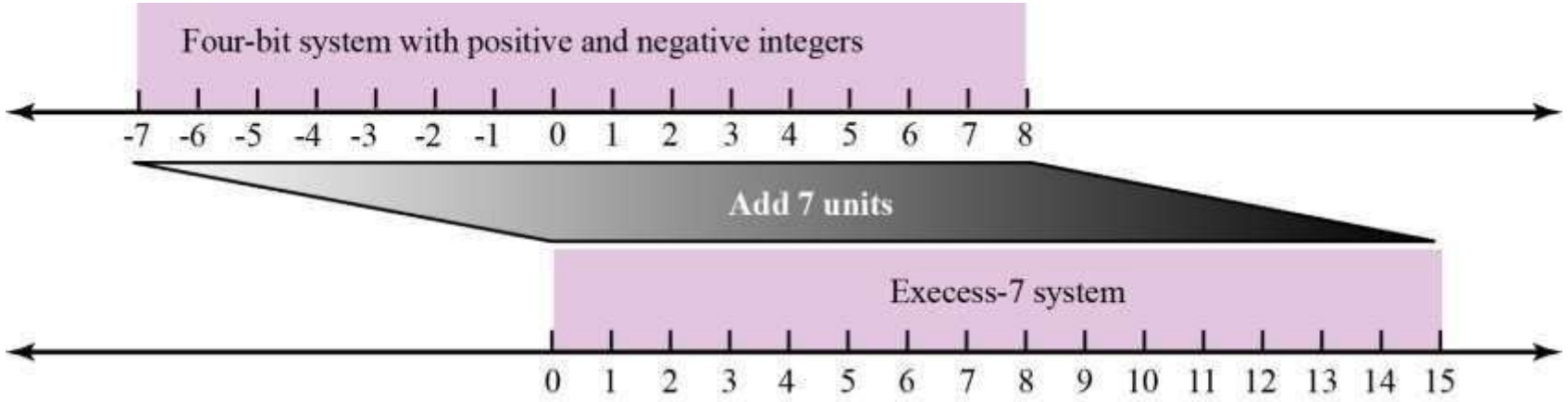
Mantissa – unsigned integer, exponent – signed integer

- Exponent - Аравтын бутархайн таслалыг зүүн буюу баруун тийш хэдэн бит шилжүүлэхийг харуулдаг тэмдэгтэй тоо юм. Хэдий бид 2-тын гүйцээлт ашиглан хадгалж болох байсан ч оронд нь илүүдэл тооцох систем гэж нэрлэгддэг шинэ илэрхийллийг ашиглаж бас болно.

- Илүүдэл тооцох системд эерэг ба сөрөг бүхэл тоо хоёулаа тэмдэггүй бүхэл тоо хэлбэрээр хадгалагддаг. Эерэг эсвэл сөрөг бүхэл тоог илэрхийлэхийн тулд сөрөг бүх тал руу жигд шилжүүлэхийн тулд эерэг бүхэл тоог (ө.х хэвийсэн гэж нэрлэдэг) нэмнэ. Энэ хэвийх утгын утга нь $2^{m-1} - 1$ бөгөөд m нь экспонент хадгалах санах ойн байршлын хэмжээ юм.

Компьютер дахь өгөгдөлийн хадгалалт

Бид 16 гэсэн бүхэл тоог 4 битээр 2т-ын тооллын системд илэрхийлж болно. Энэ муж дахь бүхэл тоо бүрт долоон нэгж /”111” хоёртоор/ нэмж оруулснаар бид бүхэл тоонуудыг баруун тийш жигд шилжүүлж, бүхэл тоонуудыг бие биетэйгээ харьцуулсан байрлалыг өөрчлөхгүйгээр бүгдийг нь эерэг болгож чадна. Энэ шинэ системийг Excess-7 гэж нэрлэдэг.



Зураг 3.11 **Excess илэрхийлэл дэх шилжилт**

Компьютер дахь өгөгдөлийн хадгалалт

5.75 гэсэн аравтын бутархай тоог Excess_127 (дан нарийвчлалтай) хэлбэрээр харуул.

Бодолт

a. Эерэг тоо тул S = 0.
b. 10-таас 2-т руу хөрвүүлбэл: $5.75 = (101.11)_2$.
c. Normalization: $(101.11)_2 = (1.1011)_2 \times 2^2$.
d. $E = 2 + 127 = 129 = (10000001)_2$, M = 1011. M-ийн баруун талд арван есийг нэмж 23 бит болгох хэрэгтэй.
e. Илэрхийллийг доор үзүүлэв:

0	10000001	10110000000000000000000
S	E	M

Энэ тоог компьютерт дараах байдлаар хадгална

01000000110110000000000000000000

Компьютер дахь өгөгдөлийн хадгалалт

Аравтын бутархай -161.875 гэсэн нарийвчлалтай) –аар дүрслэн харуул.

Бодолт

a. Тэмдэг сөрөг тул S = 1.
b. Аравтын хоёртын хувиргалт: $161.875 = (10100001.111)_2$.
c. Normalization: $(10100001.111)_2 = (1.0100001111)_2 \times 2^7$.
d. $E = 7 + 127 = 134 = (10000110)_2$ and $M = (0100001111)_2$.
e. Илэрхийлэл:

1	10000110	01000011110000000000000
S	E	M

Энэ тоог компьютерт дараах байдлаар хадгална

11000011010000111100000000000000

Жишээ 3.26

2-тын битийн загвар (11001010000000000111000100001111) нь Excess_127 форматаар хадгалагдана. Аравтын бутархай утгыг харуулна уу.

Бодоль

a. Эхний бит нь S, дараагийн найман бит, E ба үлдсэн 23 бит M-ийг илэрхийлнэ.

S	E	M
1	10010100	00000000111000100001111

b. Тэмдэг сөрөг байна.
c. Шилжүүлэгч = E - 127 = 148 - 127 = 21.
d. Энэ нь бидэнд (1.00000000111000100001111) × 221 болно.
e. Хоёртын тоо нь (10000000111000100011.11) юм.
f. Үнэмлэхүй утга нь 2,104,378.75 байна.
g. Энэ тоо нь −2,104,378.75 байна.

Компьютер дахь өгөгдлийн хадгалалт

Текстийг хадгалах

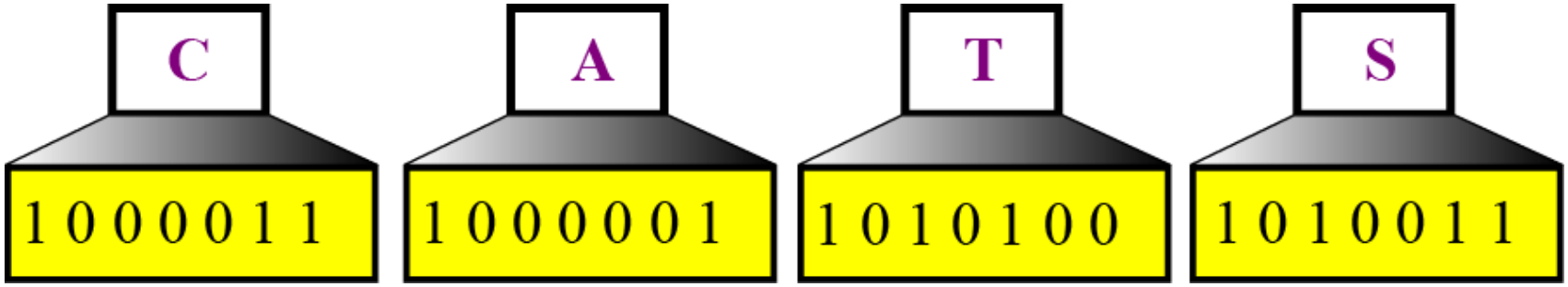
Аливаа хэл дээрх текстийн хэсэг нь тухайн хэл дээрх санааг илэрхийлэхэд ашиглагддаг тэмдгүүдийн дараалал юм.

Жишээлбэл, Англи хэл том үсгийг илэрхийлэхдээ 26 тэмдэг (A, B, C,…, Z), жижиг үсгийг илэрхийлэх 26 тэмдэг (a, b, c,…, z), есөн цифр (0, 1, 2,…, 9) цэг таслалыг илэрхийлэх тоон тэмдэгтүүд ба тэмдгүүдийг (.,?,:,;,…,!)

Дүрслэх. Хоосон, шинэ мөр, таб гэх мэт бусад тэмдгүүдийг текстийн уялдуулах, уншихад ашигладаг.

Компьютер дахь өгөгдлийн хадгалалт

Бид тэмдэг бүрийг битийн загвараар илэрхийлж чаддаг. Өөрөөр хэлбэл, дөрвөн тэмдэгээс бүтсэн “CATS” гэх мэт текстийг дөрвөн ширхэг n-битийн хэв маягаар дүрслэх боломжтой бөгөөд битийн загвар тус бүр нь нэг тэмдгийг тодорхойлдог (Зураг 3.14).



Зураг 3.13 **Битийн загвар ашиглан тэмдэгтийг илэрхийлэх нь**

ASCII - American Standard Code for Information Interchange

- **Америкийн мэдээллийн солилцооны стандарт код** буюу ASCII нь англи хэлний цагаан толгойн дараалал дээр тулгуурласан тэмдэгтийн кодлолын схем юм.
- ASCII-гаар компьютерт, санах ойн(8 битээр) нийт 128 ширхэг ялгаатай тэмдэгтийг кодлох боломжтой байдаг. Тэмдэгтийн код 0..127 завсарт оршино.
- 0-31 хүртэлх кодыг удирдах кодууд гэдэг. Удирдах код нь ямар нэг тэмдэгт илэрхийлэхгүй (дүрслэгдэх тэмдэгт байхгүй), харин ямар нэгэн үйлдлийг төлөөлнө. Ж.нь 8 гэсэн код курсорыг зүүн гар тийш нэг шилжүүлэх, 12 гэсэн код дэлгэц цэвэрлэх үйлдлийг заах гэх мэт.
- Санах ойд тэмдэгт ASCII-кодоороо хадгалагдана. Гэтэл ASCII-код бол хүснэгтэд байрлах дугаар. Ө.х. санах ойд тэмдэгтийн хэлбэр дүрс нь хадгалагддаггүй ажээ. Компьютерт өөрт нь байдаг тэмдэг дүрслэгч (character generator) хэмээх хэрэгсэл хадгалагдсан ASCII-кодоор ямар тэмдэгт болохыг таньж компьютерын дэлгэц, эсвэл цаасан дээр хэвлэдэг байна.
- Дэлхийн олон орны цагаан толгой 256-аас их тэмдэгттэй байдаг. Ийм цагаан толгой ASCII-хүснэгтэд багтахгүй. Үүнээс улбаалан тэмдэгтийг 2 байтаар кодлох аргачлал орчин үед бий болжээ.

UNICODE

- Юникод нь дэлхийн ихэнх бичлэгийн системд хэрэглэгддэг текстүүдийг зохицуулж, хоорондын уялдаа холбоог хангаж байдаг, тооцоолон бодох аж үйлдвэрийн стандарт юм.
- Юникодын хөгжүүлэлтийг хариуцдаг ашгийн төлөө бус байгууллага болох Юникод Концорциум нь одоо оршин байгаа тэмдэгтийн кодлолын бүхий л системүүдийг өөрсдийн стандарт болох Юникод Шилжүүлэлтийн Формат буюу UTF системээр сольж, компьютерын орчинд олон хэлтэй ажиллахад гардаг хүндрэл бэрхшээлүүдийг арилгахыг зорьж буй.
- Тэдний хүчин чармайлтын үр дүнд өдгөө ихэнх компьютерын програмын нутагшуулалт Юникод стандартын дагуу хийгдэх болжээ. XML, Java програмчлалын хэл, Microsoft .NET Framework болон орчин үеийн үйлдлийн системүүд бүгд Юникодыг ашиглаж байна.
- Юникодыг олон янзын тэмдэгтийн кодлолын схемд ашиглаж болдог. Хамгийн өргөн хэрэглэгддэг тайлал нь UTF-8, UTF-16 болон хэрэглээнээс хасагдаж эхэлж буй UCS-2 болно.

https://home.unicode.org/

Компьютер дахь дуу хадгалалт

- Аудио бол дуу эсвэл хөгжмийн дүрслэл юм. Аудио нь угаасаа өнөөг хүртэл бидний авч үзсэн тоо эсвэл текстээс өөр юм .

- Текст нь тоолж болох объект (тэмдэгт)- ээс бүрдэнэ: бид текстийн тэмдэгтийн тоог тоолж болно. Текст бол тоон өгөгдлийн жишээ юм .

- Аудио тоолох боломжгүй. Аудио бол аналог өгөгдлийн жишээ юм . Тодорхой хугацаанд түүний бүх утгыг хэмжих боломжтой байсан ч гэсэн хязгааргүй олон санах ойн байршил шаардагдах тул бид эдгээрийг компьютерийн санах ойд хадгалах боломжгүй юм.

Компьютер дахь дүрс хадгалалт

Зургийг растер график ба вектор график гэсэн хоёр өөр аргыг ашиглан компьютерт хадгалдаг.

Raster graphics

Растр график (эсвэл bitmap график) -ийг бид гэрэл зураг гэх мэт аналог дүрсийг хадгалах шаардлагатай үед ашигладаг. Фото зураг нь аудио мэдээлэлтэй ижил төстэй аналог өгөгдлүүдээс бүрдэнэ. Үүний ялгаа нь өгөгдлийн өнгөний эрчим (color intensity) цаг хугацааны оронд орон зайд харилцан адилгүй байдагт оршино. Энэ нь өгөгдлөөс дээж авах ёстой гэсэн үг юм. Уг тохиолдолд дээж авахыг ихэвчлэн скан хийх (scanning) гэж нэрлэдэг. Дээжийг пиксел (зургийн элемент) гэж нэрлэд эг. Өөрөөр хэлбэл зургийг бүхэлд нь жижиг пиксел болгон хуваасан бөгөөд пиксел бүрийг нэг эрчмийн утгатай гэж үздэг.

Resolution-Нарийвчлал

Аудио дээж авахтай адил зураг скан хийхдээ бид дөрвөлжин эсвэл шугаман инч тутамд хэдэн пиксел бичихээ шийдэх хэрэгтэй. Зургийн боловсруулалт дахь сканнердах түвшинг Resolution гэж нэрлэдэг. Хэрэв **resolution** нь хангалттай өндөр байвал хүний нүд нь хуулбарласан зургуудын тасалдлыг таньж чадахгүй.

Color depth - Өнгөний тодрол, гүн

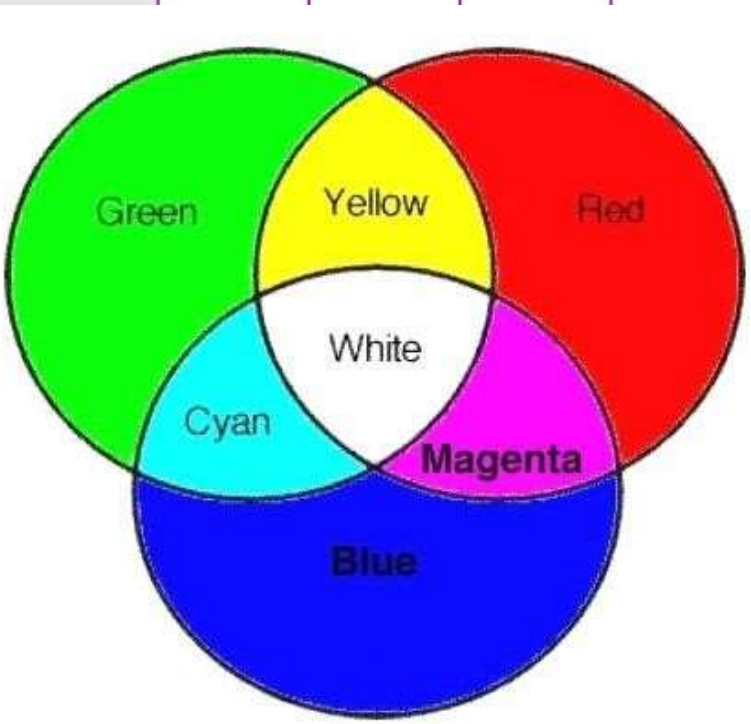
Пикселийг илэрхийлэхэд ашигладаг битүүдийн тоо, түүний өнгөний гүн нь пикселийн өнгийг өөр өөр кодчиллын техникээр хэрхэн зохицуулж байгаагаас хамаарна. Өнгөний тухай ойлголт бол бидний нүд гэрлийн туяанд хэрхэн хариу үйлдэл үзүүлэхийг хэлнэ. Бидний нүдэнд янз бүрийн фоторецептор эсүүд байдаг: зарим нь улаан, ногоон, цэнхэр (үндсэндээ RGB гэж нэрлэдэг) гэсэн гурван үндсэн өнгөнд хариу үйлдэл үзүүлдэг бол зарим нь зөвхөн гэрлийн эрчимд хариу үйлдэл үзүүлдэг байна.

True-Color Жинхэнэ өнгө

Пикселийг кодлоход ашигладаг аргуудын нэг нь True-Color гэж нэрлэгддэг бөгөөд пикселийг 24 бит ашигладаг.

Table 3.4 Some colors defined in True-Color

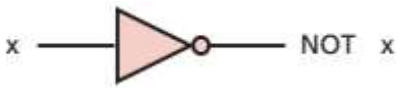
Color	*Red*	*Green*	*Blue*	*Color*	*Red*	*Green*	*Blue*
Black	0	0	0	Yell			
Red	255	0	0	Cya			
Green	0	255	0	Mag			
Blue	0	0	255	Wh			



Өгөгдлийн тооцоолол

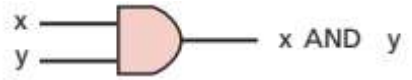
Бит нь 0 эсвэл 1 гэсэн хоёр утгын аль нэгийг авч болно. Хэрэв бид 0-ийг худал утга, 1-ыг үнэн гэж үзвэл бид Булийн алгебр дээр тодорхойлсон үйлдлүүдийг хэрэглэж болно. Жорж Булийн нэрээр нэрлэгдсэн Булийн алгебр нь логик гэж нэрлэгддэг математикийн тусгай салбар юм.

Логик үйлдлүүд ба үнэний хүснэгт



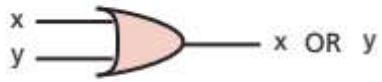
NOT

x	NOT x
0	1
1	0



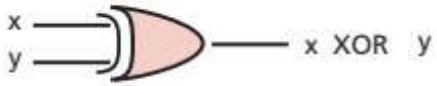
AND

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1



OR

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1



XOR

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

Логик үйлдэл

NOT

NOT оператор нь нэгдмэл оператор бөгөөд зөвхөн нэг оролтыг авдаг. Хэрэв оролт 0 бол гаралт 1, оролт 1 бол гаралт нь 0 байна. Өөрөөр хэлбэл оролтыг эсрэгээр нь эргүүлж гаргадаг гэсэн үг.

AND

AND оператор нь хоёртын оператор бөгөөд хоёр оролт авдаг. Хоёр оролт хоёулаа 1 байх нөхцөлд л гаралт нь 1 гардаг. Бусад тохиолдолд гаралт 0 байна. AND-ийн үнэний хүснэгт Оператор нь дөрвөн эгнээтэй, учир нь хоёр оролттой бол дөрвөн оролтын хослол гарна.

For $x = 0$ or 1 x AND $0 \rightarrow 0$ and 0 AND $x \rightarrow 0$

OR

OR оператор нь хоёртын оператор бөгөөд хоёр оролт авдаг. Хоёр оролтын нэг нь л 1 байхад гаралт нь 1 байна.

For $x = 0$ or 1 x OR $1 \rightarrow 1$ and 1 OR $x \rightarrow 1$

Логик үйлдэл

XOR

XOR оператор ("онцгой-эсвэл" гэж нэрлэдэг) нь мөн OR шиг хоёртын оператор юм. Хоёр оролтын нэг нь л 1 байхад үр дүн нь 1 гарна. OR операторооc ялгаатай нь бусад үед 0 үр дүн 0 гарна.

$$x \text{ XOR } y \leftrightarrow [x \text{ AND } (\text{NOT } y)] \text{ OR } [(\text{NOT } x) \text{ AND } y]$$

$$\text{For } x = 0 \text{ or } 1 \qquad 1 \text{ XOR } x \rightarrow \text{NOT } x \qquad \text{and} \qquad x \text{ XOR } 1 \rightarrow \text{NOT } x$$

Логик тооцоолол

Жишээ: AND оператор. 10011000 болон 00101010 тоонууд дээр AND үйлдэл хийж үзүүлэв.

	1	0	0	1	1	0	0	0	Input 1
AND	0	0	1	0	1	0	1	0	Input 2
	0	0	0	0	1	0	0	0	Output

Жишээ: OR оператор. 10011001 болон 00101110 тоонууд дээр OR үйдэл хийж харуулав.

	1	0	0	1	1	0	0	1	Input 1
OR	0	0	1	0	1	1	1	0	Input 2
	1	0	1	1	1	1	1	1	Output

Логик үйлдэл

Жишээ : XOR оператор. 10011001 болон 00101110 тоонуудыг XOR үйлдэл хийж үзүүлэв.

	1	0	0	1	1	0	0	1
XOR	0	0	1	0	1	1	1	0
	1	0	1	1	0	1	1	1

Input 1
Input 2
Output

Applications

Дөрвөн логик үйлдлийг битийн загварыг өөрчлөхөд ашиглаж болно.

Complementing

NOT операторын цорын ганц хэрэглээ бол бүх загварын гүйцээлтэд ашиглах явдал юм. Энэ оператор 0-1, 1-ээс 0 тутамд өөрчлөгддөг. Үүнийг заримдаа to as a one's complement operation гэж хэлдэг.

Арифметик үйлдлүүд

Арифметик үйлдлүүдэд дараах 4 үйлдэл орно.

- Нэмэх
- Хасах
- Үржих
- Хуваах

Эдгээр үйлдлийг бүхэл тоо болон бодит тоон дээр хоёуланд нь гүйцэтгэх боломжтой.

Тухайн тоог хэрхэн хадгалснаас хамааран

- Хоёртын гүйцээлт
- Sign-and-magnitude
- Floating-point хэмээн ангилж, ялгаатай форматад арифметик үйлдлүүдийг гүйцэтгэнэ.

Хоёртын гүйцээлтийг нэмэх (хасах)

Бүхэл тооны хувьд үржвэр (ноогдвор) -г нь нэмэх (хасах) үйлдлийг давтан хэрэглэснээр олох боломжтой боловч энэ нь үр өгөөж муутай байдаг. Booth procedures гэх мэт илүү үр ашигтай аргууд байдаг боловч, энэхүү номын хүрээнд зөвхөн нэмэх, хасах үйлдлийг хэлэлцэх болно. Хамгийн элбэг хэрэглэгддэг хоёртын гүйцээлтийн арифметик үйлдлийг судалъя.

Хоёртын гүйцээлт нь нэмэх болон хасах үйлдлүүд нь хоорондоо ялгаагүй байдгаараа онцлогтой. Хасах үйлдлэл уншигдах үед компьютер нь хасагчийн хоёртын гүйцээлтийг хасагдагч дээр нэмж өгдөг. Өөрөөр хэлбэл:

$A - B \leftrightarrow A + (\bar{B} + 1)$, $((\bar{B} + 1))$ нь B тооны хоёртын гүйцээлт

Хоёртын гүйцээлтийг нэмэх (хасах)

Хоёртын гүйцээлтийн тоонуудыг нэмэх нь маш энгийн бөгөөд ердийн аравтуудыг нэмж байгаатай адилаар орон орноор нь (багана баганаар нь) цифрүүдийг нь харгалзуулан нэмдэг.

Хэрэв илүү гарвал тухайн тоог өмнөх оронд санах (carry) зарчмаар явна. Хамгийн сүүлд нэмсэн баганын санасан тоог хаядаг.

Анхаарах нь: Бид компьютерт тоон дээр арифметик үйлдлүүдийг хийхдээ тоо тус бүр болон үр дүнгүүд нь битийн хуваарилалтаар тодорхойлогдсон мужид байх ёстой гэдгийг санах хэрэгтэй.

Хоёртын гүйцээлтийг нэмэх

Жишээ 4.16

А болон В хоёртын гүйцээлт бүхий бүхэл тоонууд өгөгдсөн бол нийлбэрийг ол.

$$A = (00010001)_2 \qquad B = (00010110)_2$$

Бодолт:

			1						Carry
	0	0	0	1	0	0	0	1	A
+	0	0	0	1	0	1	1	0	B
	0	0	1	0	0	1	1	1	R

Шалгах нь: (+17) + (+22) = (+39)

Sign-and-magnitude нэмэх

Жишээ J.1

Дараах бодлогын бодолтыг харуулна уу.

$$(+5.75) + (-7.0234375) = (-1.2734375)$$

Бодолт:

Дээрх тоонуудыг IEEE стандартаар илэрхийлбэл:

	S	E	M
A	0	10000001	01110000000000000000000
B	1	10000001	11000001100000000000000

Denormalization хийвэл:

	S	E	Denormalized M
A	0	10000001	01110000000000000000000
B	1	10000001	11000001100000000000000

Sign-and-magnitude нэмэх

Жишээ J.1

- Exponent –үүд нь ижил байгаа тул тэнцүүлэх шаардлагагүй ба нэмэх үйлдлийг шууд гүйцэтгэнэ.

	S	E	Denormalized M
A	0	10000001	01110000000000000000000
B	1	10000001	11000001100000000000000
R	1	10000010	00101000110000000000000

- Гарсан үр дүнг нормальчлан (normalization) exponent -д 3 удаа бууруулалт (decrement) хийн mantissa –г зүүн тийш 3 шилжүүлнэ (shift) :

	S	E	M
R	1	01111111	01000110000000000000000

АШИГЛАСАН МАТЕРИАЛ

Foundations of Computer Science, Behrouz A. Forouzan, Fourth Edition, Cengage Learning EMEA, 2018, Chapter 3, Chapter 4.

Tuyatsetseg Badarch, "Data communications and computer networking" , third edition, 2014. Chapter 4.

АНХААРАЛ ХАНДУУЛСАНД БАЯРЛАЛАА