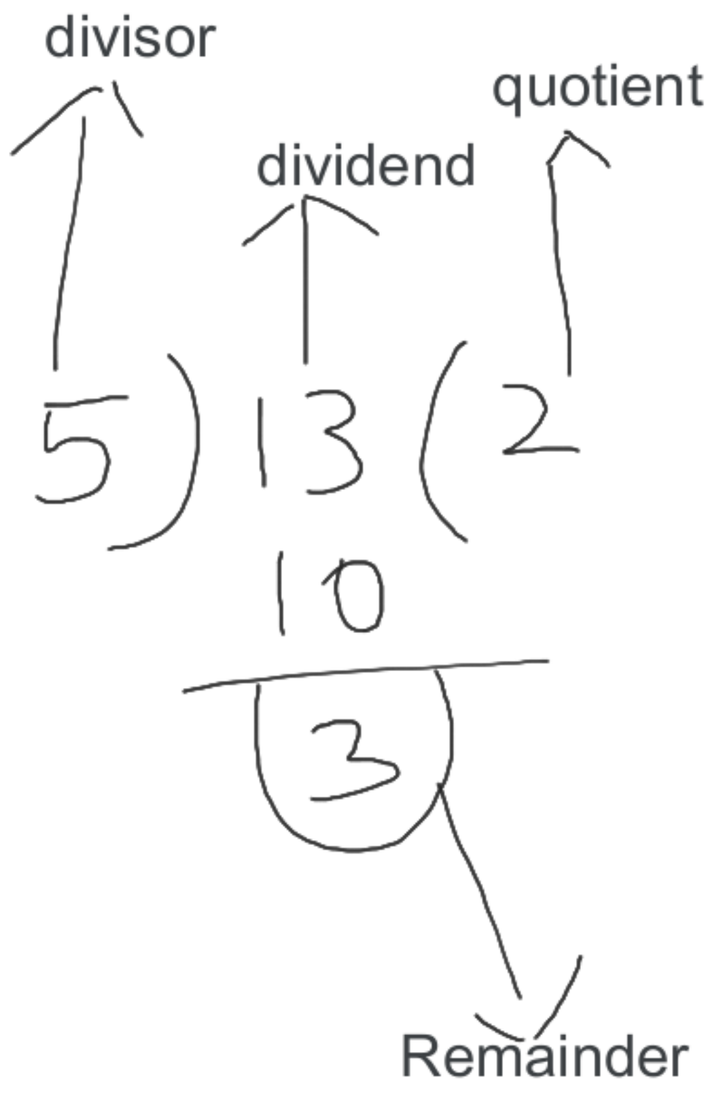
Division Instructions

Courtesy: UMBC and JBLearning

Division Instruction Formats

- `idiv` *`source`*
  for signed operands
- `div` *`source`*
  for unsigned operands
- Divisor
  - Identified by *source*
  - Byte, word, doubleword, or quadword
  - In memory or register, but **NOT immediate**
- Dividend
  - Implied based on the *source*
  - Double-length (with respect to the *source*)
  - Instructions exist to produce double-length dividend



Implicit Dividend for `div` and `idiv`

- Byte source divided into word in AX
- Word source divided into doubleword in DX:AX
- Doubleword source divided into quadword in EDX:EAX
- Quadword source divided into RDX:RAX

```
div BH----------------->AH:AL divided by BH
div CX------------------->DX:AX divided by CX
div EBX----------------->EDX:EAX divided by EBX
div RBX----------------->RDX:RAX divided by RBX
```

Results of `div` and `idiv`

- Byte-size divisor:
  quotient in AL and remainder in AH
- Word-size divisor:
  quotient in AX and remainder in DX
- Doubleword-size divisor:
  quotient in EAX and remainder in EDX
- Quadword-size divisor:
  quotient in RAX and remainder in RDX

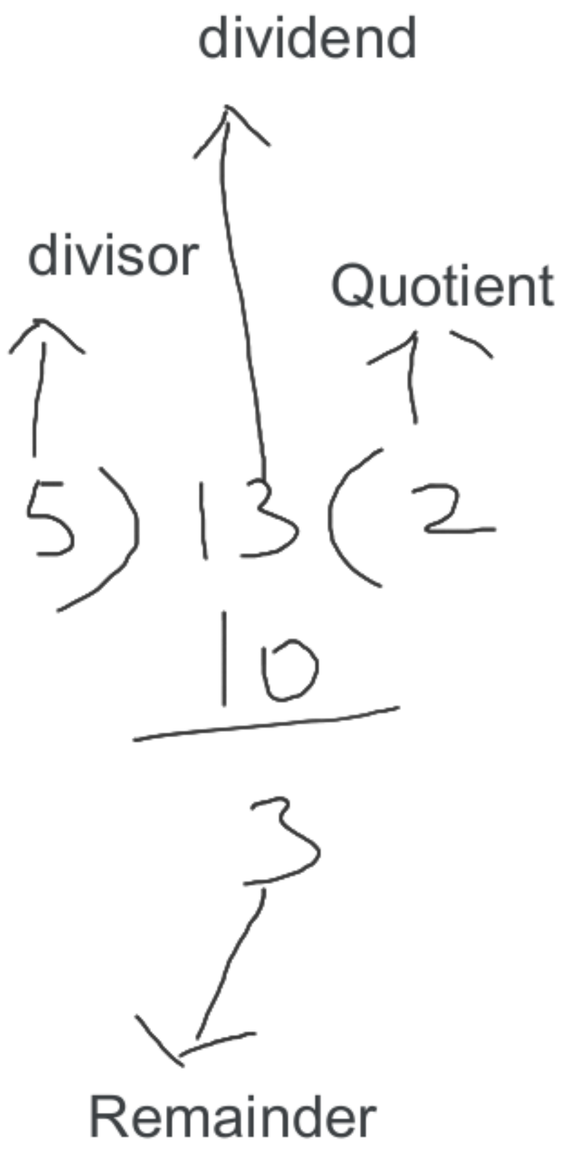
div BH----------->AH:AL divided by BH----------->quotient-->AL, Rem --->AH
div CX----------->DX:AX divided by CX----------->quotient-->AX, Rem---> Dx
div EBX----------->EDX:EAX divided by EBX----------->q-EAX,Rem-->EDX
div RBX----------->RDX:RAX divided by RBX----------->q-RAX, Rem-->RDX

Results of `div` and `idiv`

- All division operations satisfy the relation dividend = quotient*divisor + remainder
- For signed division
  - The sign of the quotient : standard sign rules.
  - The sign of the remainder : same sign as dividend.

div, idiv Instruction Operation

divisor length	dividend	Quotient	Remainder
byte	AX	AL	AH
word	DX:AX	AX	DX
double word	EDX:EAX	EAX	EDX
quad word	RDX:RAX	RAX	RDX




```
div bh
div cx
div ecx
div rbx
```

Flag Settings

- Division instructions do not set flags to any meaningful values.
- They may change previously set values of AF, CF, OF, PF, SF, or ZF.

Unsigned Division Example

- *Before*
  EDX: 00 00 00 00
  EAX: 00 00 00 64
  EBX: 00 00 00 0D
- *Instruction*
  `div ebx ; 100/13`
- *After*
  EDX: 00000009
  EAX: 00000007

100 = 7 * 13 + 9

100

13

100/13----->
q-7
R-9

div ebx
ebx------>double word size register
double word size reg---->dividend---->EDX:EAX
quotient---->EAX, Remainder---->EDX

Signed Division Example

- *Before*
  EDX: FF FF FF FF
  EAX: FF FF FF 9C
  ECX: 00 00 00 0D

-100

13

- *Instruction*

```
idiv  ecx  ; -100/13
```

-100/13

- *After*
  EDX: FFFFFFF7
  EAX: FFFFFFF9

Q->-7
R->-9

$-100 = (-7) * 13 + (-9)$

Errors in Division

- Caused by
  - Dividing by 0, or
  - Quotient too large to fit in destination
- Triggers an *exception*
  - The interrupt handler routine that services this exception may vary from system to system.
  - When a division error occurs for a program running under Visual Studio, an error window pops up.

```
mov ebx,0
div ebx
```

ModR/M->Reg Field spec

	000	001	010	011	100	101	110	111
80,81	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
D0,D1	ROL	ROR	RCL	RCR	SHL	SHR		SAR
F6,F7	TEST		NOT	NEG	MUL	IMUL	DIV	IDIV
FE,FF	INC	DEC					PUSH	

div, idiv Instruction encoding

Operand	Opcode	Obj Code: Byte Length
register 8	F6	2
register 16	F7	3
register 32	F7	2
register 64	F7	3
memory byte	F6	2+
memory word	F7	3+
memory doubleword	F7	2+
Memory quadword	F7	3+

- Applied Prefix: 66 (word) and 4x (for 64-

div, idiv Instruction encoding

```
00000000  F6 F4          div AH
00000002  41/ F6 F8          idiv R8B
00000005  66| F7 F3          div BX
00000008  66| F7 FD          idiv BP
0000000B  F7 F1          div ECX
0000000D  41/ F7 FF          idiv R15D
00000010  48/ F7 F3          div RBX
00000013  49/ F7 FC          idiv R12
```

1	0	0	1	1	1	1	1	1	0	0

1100 => R12

idiv

```
00000032  F6 35 000001CE R div bytenum
00000038  F6 3B          idiv BYTE PTR [RBX]
0000003A  66| F7 3D          idiv wnum
      000001CF R
00000041  66| F7 31          div WORD PTR [RCX]
00000044  F7 3D 000001D1 R idiv dwnum
0000004A  F7 36          div DWORD PTR [RSI]
0000004C  48/ F7 35          div qwnum
      000001D5 R
00000053  48/ F7 3F          idiv QWORD PTR [RDI]
```

- Applied Prefix: 66 (word) and 4x (for 64-bit specific).

Preparing for Division

- Often dividend must be extended to double length
- Example
  - Copy a doubleword dividend to EAX
  - Extend dividend to EDX:EAX
    - For unsigned division, use `mov edx, 0`
    - For signed division, use `cdq` instruction
  - Finally use `div` or `idiv` instruction

if we want to do divide FFFFABCD by AB7C

Mov EAX,0FFFFABCDH

Mov ECX,00000AB7CH

div ECX

edx:eax divided by ECX

in this case edx have some random value

Convert Instructions

- They require NO operand
- `cbw` – sign extends the byte in AL to the word in AX
- `cwd` – sign extends the word in AX to the doubleword in DX:AX
- `cdq` – sign extends the doubleword in EAX to the quadword in EDX:EAX
- `cqo` – sign extends the quadword in RAX to RDX:RAX

question1
before :
eax=FFFFC1A2
double word value= FFFFFFFA

instruction : mul value → value*eax----> result is stored in edx:eax

Answer:

```
.data
value dword 0FFFFFFFAH

.code
mov eax,0FFFFC1A2H

mul value
```