# Extended Array Example – lowercase conversion

COS10004 Lecture 9.5

### Array (and ASCII) example – convert to lowercase

- Let's write an ASM program that reads a string from input and converts all alphabetic characters to lower case
- Algorithm:
  - Store input string in array
  - Iterate through the array and check each character
    - If character needs converting, convert to lowercase
    - If not, leave as is
  - Once complete, write string to output

#### Using ASCII to do conversion

- Recall every character has an ASCII code (0 - 255)
- Uppercase alphabetic characters are within the range: 65 - 90

char	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec
•	140	60	96	@	100	40	64	space	040	20	32	NULL	000	0	0
а	141	61	97	Α	101	41	65	1	041	21	33	SOH	001	1	1
b	142	62	98	В	102	42	66	.10	042	22	34	STX	002	2	2
C	143	63	99	С	103	43	67	#	043	23	35	ETX	003	3	3
d	144	64	100	D	104	44	68	\$	044	24	36	EOT	004	4	4
е	145	65	101	E	105	45	69	%	045	25	37	ENQ	005	5	5
f	146	66	102	F	106	46	70	&	046	26	38	ACK	006	6	6
g	147	67	103	G	107	47	71		047	27	39	BEL	007	7	7
h	150	68	104	н	110	48	72	(	050	28	40	BS	010	8	8
i	151	69	105	1	111	49	73	)	051	29	41	TAB	011	9	9
j	152	6a	106	J	112	4a	74	*	052	2a	42	LF	012	a	10
k	153	6b	107	K	113	4b	75	+	053	2b	43	VT	013	b	11
1	154	6c	108	L	114	4c	76	,	054	2c	44	FF	014	С	12
m	155	6d	109	M	115	4d	77		055	2d	45	CR	015	d	13
n	156	6e	110	N	116	4e	78	*:	056	2e	46	so	016	e	14
0	157	6f	111	0	117	4f	79	1	057	2f	47	SI	017	f	15
р	160	70	112	P	120	50	80	0	060	30	48	DLE	020	10	16
q	161	71	113	Q	121	51	81	1	061	31	49	DC1	021	11	17
r	162	72	114	R	122	52	82	2	062	32	50	DC2	022	12	18
S	163	73	115	S	123	53	83	3	063	33	51	DC3	023	13	19
t	164	74	116	Т	124	54	84	4	064	34	52	DC4	024	14	20
u	165	75	117	U	125	55	85	5	065	35	53	NAK	025	15	21
v	166	76	118	V	126	56	86	6	066	36	54	SYN	026	16	22
w	167	77	119	w	127	57	87	7	067	37	55	ETB	027	17	23
x	170	78	120	X	130	58	88	8	070	38	56	CAN	030	18	24
У	171	79	121	Υ	131	59	89	9	071	39	57	EM	031	19	25
z	172	7a	122	Z	132	5a	90	:	072	3a	58	SUB	032	1a	26
{	173	7b	123		133	5b	91	,	073	3b	59	ESC	033	1b	27
i	174	7c	124	Í	134	5c	92	<	074	Зс	60	FS	034	1c	28
}	175	7d	125	i	135	5d	93	=	075	3d	61	GS	035	1d	29
~	176	7e	126	٨	136	5e	94	>	076	3e	62	RS	036	1e	30
DEL	177	7f	127		137	5f	95	?	077	3f	63	US	037	1f	31

#### Using ASCII to do conversion

- Recall every character has an ASCII code (0 - 255)
- Uppercase alphabetic characters are within the range: 65 – 90
- Lowercase alphabetic characters are within the range: 97 – 122

cha	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec
,	140	60	96	@	100	40	64	space	040	20	32	NULL	000	0	0
а	141	61	97	A	101	41	65	1	041	21	33	SOH	001	1	1
b	142	62	98	В	102	42	66	· m	042	22	34	STX	002	2	2
C	143	63	99	C	103	43	67	#	043	23	35	ETX	003	3	3
d	144	64	100	D	104	44	68	\$	044	24	36	EOT	004	4	4
е	145	65	101	E	105	45	69	%	045	25	37	ENQ	005	5	5
f	146	66	102	F	106	46	70	&	046	26	38	ACK	006	6	6
g	147	67	103	G	107	47	71	1	047	27	39	BEL	007	7	7
h	150	68	104	н	110	48	72	(	050	28	40	BS	010	8	8
i	151	69	105	1	111	49	73	)	051	29	41	TAB	011	9	9
j	152	6a	106	J	112	4a	74	*	052	2a	42	LF	012	a	10
k	153	6b	107	K	113	4b	75	+	053	2b	43	VT	013	b	11
1	154	6c	108	L	114	4c	76	,	054	2c	44	FF	014	С	12
m	155	6d	109	M	115	4d	77		055	2d	45	CR	015	d	13
n	156	6e	110	N	116	4e	78	*:	056	2e	46	so	016	e	14
0	157	6f	111	0	117	4f	79	1	057	2f	47	SI	017	f	15
р	160	70	112	P	120	50	80	0	060	30	48	DLE	020	10	16
q	161	71	113	Q	121	51	81	1	061	31	49	DC1	021	11	17
r	162	72	114	R	122	52	82	2	062	32	50	DC2	022	12	18
S	163	73	115	S	123	53	83	3	063	33	51	DC3	023	13	19
t	164	74	116	Т	124	54	84	4	064	34	52	DC4	024	14	20
u	165	75	117	U	125	55	85	5	065	35	53	NAK	025	15	21
v	166	76	118	V	126	56	86	6	066	36	54	SYN	026	16	22
w	167	77	119	w	127	57	87	7	067	37	55	ETB	027	17	23
×	170	78	120	X	130	58	88	8	070	38	56	CAN	030	18	24
У	171	79	121	Y	131	59	89	9	071	39	57	EM	031	19	25
z	172	7a	122	Z	132	5a	90	:	072	3a	58	SUB	032	1a	26
{	173	7b	123	[	133	5b	91	,	073	3b	59	ESC	033	1b	27
ì	174	7c	124	i	134	5c	92	<	074	Зс	60	FS	034	1c	28
3	175	7d	125	i	135	5d	93	=	075	3d	61	GS	035	1d	29
~	176	7e	126	^	136	5e	94	>	076	3е	62	RS	036	1e	30
DE	177	7f	127		137	5f	95	?	077	3f	63	US	037	1f	31

#### Using ASCII to do conversion

- Recall every character has an ASCII code (0 - 255)
- Uppercase alphabetic characters are within the range: 65 – 90
- Lowercase alphabetic characters are within the range: 97 – 122
- Conversion of uppercase to lowercase requires subtracting 32 from ASCII value
  - E.g. "A" (65) -> (65 + 32) -> "a" (97)

char	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec
•	140	60	96	@	100	40	64	space	040	20	32	NULL	000	0	0
а	141	61	97	A	101	41	65	1	041	21	33	SOH	001	1	1
b	142	62	98	В	102	42	66	H	042	22	34	STX	002	2	2
c	143	63	99	C	103	43	67	#	043	23	35	ETX	003	3	3
d	144	64	100	D	104	44	68	\$	044	24	36	EOT	004	4	4
е	145	65	101	E	105	45	69	%	045	25	37	ENQ	005	5	5
f	146	66	102	F	106	46	70	&	046	26	38	ACK	006	6	6
g	147	67	103	G	107	47	71		047	27	39	BEL	007	7	7
h	150	68	104	н	110	48	72	(	050	28	40	BS	010	8	8
i	151	69	105	1	111	49	73	)	051	29	41	TAB	011	9	9
j	152	6a	106	J	112	4a	74	*	052	2a	42	LF	012	a	10
k	153	6b	107	K	113	4b	75	+	053	2b	43	VT	013	b	11
1	154	6c	108	L	114	4c	76	,	054	2c	44	FF	014	С	12
m	155	6d	109	M	115	4d	77		055	2d	45	CR	015	d	13
n	156	6e	110	N	116	4e	78	*:	056	2e	46	so	016	e	14
0	157	6f	111	0	117	4f	79	1	057	2f	47	SI	017	f	15
р	160	70	112	P	120	50	80	0	060	30	48	DLE	020	10	16
q	161	71	113	Q	121	51	81	1	061	31	49	DC1	021	11	17
r	162	72	114	R	122	52	82	2	062	32	50	DC2	022	12	18
S	163	73	115	S	123	53	83	3	063	33	51	DC3	023	13	19
t	164	74	116	T	124	54	84	4	064	34	52	DC4	024	14	20
u	165	75	117	U	125	55	85	5	065	35	53	NAK	025	15	21
v	166	76	118	V	126	56	86	6	066	36	54	SYN	026	16	22
w	167	77	119	W	127	57	87	7	067	37	55	ETB	027	17	23
×	170	78	120	X	130	58	88	8	070	38	56	CAN	030	18	24
У	171	79	121	Y	131	59	89	9	071	39	57	EM	031	19	25
z	172	7a	122	Z	132	5a	90	:	072	3a	58	SUB	032	1a	26
{	173	7b	123	[	133	5b	91	;	073	3b	59	ESC	033	1b	27
1	174	7c	124	1	134	5c	92	<	074	3с	60	FS	034	1c	28
}	175	7d	125	1	135	5d	93	=	075	3d	61	GS	035	1d	29
~	176	7e	126	٨	136	5e	94	>	076	3e	62	RS	036	1e	30
DEL	177	7f	127		137	5f	95	?	077	3f	63	US	037	1f	31

• First we need to define an array to store the string:

```
.Align 256
stringData: .BLOCK 256
```

We then need to read an input string into the array:

```
MOV RO, #stringData
STR RO, .ReadString
.Align 256
stringData: .BLOCK 256
```

- We now need to implement a loop that inspects each character.
- Lets start with the basic loop structures first. We need:
  - A register to hold the array base address
  - A register to hold the index of the current character
  - A register to store the actual character value once
  - A condition to determine when we have finished scanning each character

- We now need to implement a loop that inspects each character.
- Lets start with the basic loop structures first. We need:
  - A Label to mark the start of the loop (so we can branch back)
  - A register to hold the array base address (we already have this in RO) MOV RO, #stringData
  - A register to hold the index of the current character (let's use R1)

    MOV R1, #0 // initialise index to 0
  - A register to store the actual character value once (let's use R2)

```
LDRB R2, [R0 + R1] // reads the byte at address R0+R1
```

• A condition to determine when we have finished scanning each character

• we can look for the "NULL" character, which is always appended to the end of any

input string.

char	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec	char	oct	hex	dec
	140	60	96	@	100	40	64	space	040	20	32	NULL	000	0	0
а	141	61	97	Α	101	41	65	1	041	21	33	SOH	001	1	1
b	142	62	98	В	102	42	66	No.	042	22	34	STX	002	2	2
C	143	63	99	C	103	43	67	#	043	23	35	ETX	003	3	3
d	144	64	100	D	104	44	68	\$	044	24	36	EOT	004	4	4
e	145	65	101	E	105	45	69	%	045	25	37	ENQ	005	5	5
f	146	66	102	F	106	46	70	&	046	26	38	ACK	006	6	6
g	147	67	103	G	107	47	71	,	047	27	39	BEL	007	7	7
h	150	68	104	Н	110	48	72	(	050	28	40	BS	010	8	8
i	151	69	105	1	111	49	73	)	051	29	41	TAB	011	9	9
j	152	6a	106	J	112	4a	74	*	052	2a	42	LF	012	a	10
k	153	6b	107	K	113	4b	75	+	053	2b	43	VT	013	b	11
1	154	6c	108	L	114	4c	76	,	054	2c	44	FF	014	С	12
m	155	6d	109	M	115	4d	77		055	2d	45	CR	015	d	13
n	156	6e	110	N	116	4e	78	*:	056	2e	46	so	016	e	14
0	157	6f	111	0	117	4f	79	1	057	2f	47	SI	017	f	15
р	160	70	112	P	120	50	80	0	060	30	48	DLE	020	10	16
q	161	71	113	Q	121	51	81	1	061	31	49	DC1	021	11	17
r	162	72	114	R	122	52	82	2	062	32	50	DC2	022	12	18
s	163	73	115	S	123	53	83	3	063	33	51	DC3	023	13	19
t	164	74	116	T	124	54	84	4	064	34	52	DC4	024	14	20
u	165	75	117	U	125	55	85	5	065	35	53	NAK	025	15	21
V	166	76	118	V	126	56	86	6	066	36	54	SYN	026	16	22
w	167	77	119	W	127	57	87	7	067	37	55	ETB	027	17	23
×	170	78	120	X	130	58	88	8	070	38	56	CAN	030	18	24
У	171	79	121	Y	131	59	89	9	071	39	57	EM	031	19	25
z	172	7a	122	Z	132	5a	90	:	072	3a	58	SUB	032	1a	26
{	173	7b	123	[	133	5b	91	;	073	3b	59	ESC	033	1b	27
- 1	174	7c	124	1	134	5c	92	<	074	Зс	60	FS	034	1c	28
}	175	7d	125	1	135	5d	93	=	075	3d	61	GS	035	1d	29
~	176	7e	126	۸	136	5e	94	>	076	3e	62	RS	036	1e	30
DEL	177	7f	127		137	5f	95	?	077	3f	63	US	037	1f	31

A condition to determine when we have finished scanning each character

• we can look for the "NULL" character, which is always appended to the end of any

input string.

	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
	0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	•
	1	1	001	SOH	33	21	041	1	65	41	101	Α	97	61	141	a
	2	2	002	STX	34	22	042	m	66	42	102	В	98	62	142	b
	3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	C
The NULL character has ASCII value 0	4	4	004	EOT	36	24	044	\$	68	44	104	D	100	64	144	d
THE NULL CHaracter has ASCII value of	5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	е
	6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
	7	7	007	BEL	39	27	047	,	71	47	107	G	103	67	147	g
	8	8	010	BS	40	28	050	(	72	48	110	Н	104	68	150	h
	9	9	011	TAB	41	29	051	)	73	49	111	1	105	69	151	i
	10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
	11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
	12	С	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	1
	13	d	015	CR	45	2d	055		77	4d	115	M	109	6d	155	m
	14	e	016	SO	46	2e	056	*:	78	4e	116	N	110	6e	156	n
	15	f	017	SI	47	2f	057	1	79	4f	117	0	111	6f	157	0
	16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	р
	17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
	18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
	19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	S
	20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
	21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
	22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	V
	23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
	24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	X
	25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	У
	26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
	27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
	28	1c	034	FS	60	Зс	074	<	92	5c	134	1	124	7c	174	1
	29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
	30	1e	036	RS	62	3e	076	>	94	5e	136	٨	126	7e	176	~
	31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL
														Www	.alpharit	chms.com

- A condition to determine when we have finished scanning each character
  - we can look for the "NULL" character, which is always appended to the end of any input string.
  - So ....

CMP R2, #0
BNE toLower

- A condition to determine when we have finished scanning each character
  - we can look for the "NULL" character, which is always appended to the end of any input string.
  - So ....

```
CMP R2, #0
BNE toLower
```

This will keep iterating through the loop "toLower" until the NULL character is reached

## Uppercase to lowercase - putting loop together

```
MOV RO, #stringData
STR RO, .ReadString
                       //index
MOV R1, #0
                       // start of loop
toLower:
                       // read byte from array
LDRB R2, [R0+R1]
                     // increment index
ADD R1, R1, #1
CMP R2, #0
BNE toLower
.Align 256
stringData: .BLOCK 256
```

#### Uppercase to lowercase - conversion code

- We now need to consider the conversion:
  - Only need to convert if read character is actually uppercase alphabetic
    - Convert only if ASCII value is between 65 and 90
  - If character is uppercase alphabetic:
    - Add 32 to the value
    - Write the new value back to the array
  - If character is not uppercase alphabetic:
    - Leave as is (i.e. skip the above)

#### Uppercase to lowercase - conversion code

- We now need to consider the conversion:
  - Only need to convert if read character is actually uppercase alphabetic
    - Convert only if ASCII value is between 65 and 90
  - If character is uppercase alphabetic:
    - Add 32 to the value
    - Write the new value back to the array
  - If character is not uppercase alphabetic:
    - Leave as is (i.e. skip the above)

#### Determine if its uppercase alphabetic

```
// start of loop
toLower:
LDRB R2, [R0+R1]
                           // read byte from array
CMP R2, #65
                           // check lower value limit
BLT skipConversion
CMP R2, #90
                           // check upper value limit
BGT skipConversion
skipConversion:
                            // jump to here if value not uppercase
ADD R1, R1, #1
                           // increment index
CMP R2, #0
BNE toLower
```

#### if it's uppercase, convert to lowercase

```
// start of loop
toLower:
LDRB R2, [R0+R1]
                               // read byte from array
CMP R2, #65
                                // check lower value limit
BLT skipConversion
CMP R2, #90
                               // check upper value limit
BGT skipConversion
ADD R2, R2, #32
                                // convert to lowercase value
STRB R2, [R0+R1]
                                // write it back to the array
skipConversion:
                       // jump to here if value not uppercase
ADD R1, R1, #1
                                // increment index
CMP R2, #0
BNE toLower
```

#### All together

```
MOV RO, #stringData
STR RO, . ReadString
MOV R1, #0
toLower:
          LDRB R2, [R0+R1]
          CMP R2, #65
          BLT skipConversion
          CMP R2, #90
          BGT skipConversion
          ADD R2, R2, #32
          STRB R2, [R0+R1]
          skipConversion:
          ADD R1, R1, #1
          CMP R2, #0
          BNE toLower
STR RO, .WriteString
HALT
.Align 256
stringData: .BLOCK 128
```

```
// store array base address
// read in input
// initialise index
// start of loop
// read byte from array
// check lower value limit
// if lower than uppercase range, skip conversion
// check upper value limit
// if greater than uppercase range, skip conversion
// convert to lowercase value
// write it back to the array
// branch destination if value was not uppercase
// increment index
// check if NULL character
// Keep looping if not NULL character
// write converted string to output
```