```
#/* $begin ncopy-ys */
# 948bytes < 1000bytes
# Average CPE = 9.89
# The trick used in the ncopy function is loop unrolling.
# 1.Fisrt we iteratively 'ncopy' 16 elements until there are fewer
# than 16 elements left.
# 2.Then check if left elements are more than 8, and if so,
# we 'ncopy' 8 elements.
# 3.Repeat procedure2 by checking and 'ncopy' 4, 2, 1 element(s).
# The modifications above reduces number of condition branch
# instructions, and thus improve the performance of CPU.
# Also, some sequences of instructions are modified.
# Src-plus and count-plus instructions are inserted between
# some mrmovl and rmmovl instructions to avoid a stalling
# due to data hazard.
##################################################################
# ncopy.ys - Copy a src block of len ints to dst.
# Return the number of positive ints (>0) contained in src.
#
# Include your name and ID here.
#
# Describe how and why you modified the baseline code.
#
##################################################################
# Do not modify this portion
# Function prologue.
ncopy:     pushl %ebp            # Save old frame pointer
       rrmovl %esp,%ebp  # Set up new frame pointer
       pushl %esi            # Save callee-save regs
       pushl %ebx
       pushl %edi
       mrmovl 8(%ebp),%ebx    # src
       mrmovl 16(%ebp),%edx # len
       mrmovl 12(%ebp),%ecx  # dst

##################################################################
# You can modify this portion
xorl %eax,%eax             # count = 0;

############## Iteratively 'ncopy' 16 elements until #############
############## fewer than 16 elements left.              #############
Loop5:     iaddl    $-16,%edx   # len-=16 > 0 ?
       jl Loop4          # if so, goto Loop:
```

```
        mrmovl (%ebx), %esi      # read val from src...
        andl %esi, %esi          # val <= 0?
        jle Npos51               # if so, goto Npos:
        iaddl    $1, %eax        # count++
Npos51:  rmmovl %esi, (%ecx)     # ...and store it to dst

        mrmovl 4(%ebx), %esi     # read val from src...
        andl %esi, %esi          # val <= 0?
        jle Npos52               # if so, goto Npos:
        iaddl    $1, %eax        # count++
Npos52:  rmmovl %esi, 4(%ecx)    # ...and store it to dst

        mrmovl 8(%ebx), %esi     # read val from src...
        andl %esi, %esi          # val <= 0?
        jle Npos53               # if so, goto Npos:
        iaddl    $1, %eax        # count++
Npos53:  rmmovl %esi, 8(%ecx)    # ...and store it to dst

        mrmovl 12(%ebx), %esi    # read val from src...
        andl %esi, %esi          # val <= 0?
        jle Npos54               # if so, goto Npos:
        iaddl    $1, %eax        # count++
Npos54:  rmmovl %esi, 12(%ecx)   # ...and store it to dst

        mrmovl 16(%ebx), %esi    # read val from src...
        andl %esi, %esi          # val <= 0?
        jle Npos55               # if so, goto Npos:
        iaddl    $1, %eax        # count++
Npos55:  rmmovl %esi, 16(%ecx)   # ...and store it to dst

        mrmovl 20(%ebx), %esi    # read val from src...
        andl %esi, %esi          # val <= 0?
        jle Npos56               # if so, goto Npos:
        iaddl    $1, %eax        # count++
Npos56:  rmmovl %esi, 20(%ecx)   # ...and store it to dst

        mrmovl 24(%ebx), %esi    # read val from src...
        andl %esi, %esi          # val <= 0?
        jle Npos57               # if so, goto Npos:
        iaddl    $1, %eax        # count++
Npos57:  rmmovl %esi, 24(%ecx)   # ...and store it to dst

        mrmovl 28(%ebx), %esi    # read val from src...
        andl %esi, %esi          # val <= 0?
```

```
        jle Npos58              # if so, goto Npos:
        iaddl    $1, %eax       # count++
Npos58:   rmmovl %esi, 28(%ecx)   # ...and store it to dst

        mrmovl 32(%ebx), %esi   # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos59              # if so, goto Npos:
        iaddl    $1, %eax       # count++
Npos59:   rmmovl %esi, 32(%ecx)   # ...and store it to dst

        mrmovl 36(%ebx), %esi   # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos510             # if so, goto Npos:
        iaddl    $1, %eax       # count++
Npos510:rmmovl %esi, 36(%ecx)   # ...and store it to dst

        mrmovl 40(%ebx), %esi   # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos511             # if so, goto Npos:
        iaddl    $1, %eax       # count++
Npos511:rmmovl %esi, 40(%ecx)   # ...and store it to dst

        mrmovl 44(%ebx), %esi   # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos512             # if so, goto Npos:
        iaddl    $1, %eax       # count++
Npos512:rmmovl %esi, 44(%ecx)   # ...and store it to dst

        mrmovl 48(%ebx), %esi   # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos513             # if so, goto Npos:
        iaddl    $1, %eax       # count++
Npos513:rmmovl %esi, 48(%ecx)   # ...and store it to dst

        mrmovl 52(%ebx), %esi   # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos514             # if so, goto Npos:
        iaddl    $1, %eax       # count++
Npos514:rmmovl %esi, 52(%ecx)   # ...and store it to dst

        mrmovl 56(%ebx), %esi   # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos515             # if so, goto Npos:
        iaddl    $1, %eax       # count++
```

```
Npos515:rmmovl %esi, 56(%ecx)   # ...and store it to dst

        mrmovl 60(%ebx), %esi  # read val from src...
        iaddl    $64,%ebx           # src+=16
        rmmovl %esi, 60(%ecx)  # ...and store it to dst
        andl %esi, %esi             # val <= 0?
        jle Npos516            # if so, goto Npos:
        iaddl    $1, %eax           # count++
Npos516:iaddl    $64,%ecx           # dst+=16
        jmp Loop5              # goto Loop:
########## Iterative 'ncopy' of 16 elements is over ############



### Check if left elements are more than 8, and if so, ########
### we 'ncopy' 8 elements.                ################
Loop4:    iaddl    $8,%edx            # len-=4 > 0 ?
        jl Loop3          # if so, goto Loop:

        mrmovl (%ebx), %esi      # read val from src...
        andl %esi, %esi             # val <= 0?
        jle Npos41             # if so, goto Npos:
        iaddl    $1, %eax           # count++
Npos41:   rmmovl %esi, (%ecx)      # ...and store it to dst

        mrmovl 4(%ebx), %esi    # read val from src...
        andl %esi, %esi             # val <= 0?
        jle Npos42             # if so, goto Npos:
        iaddl    $1, %eax           # count++
Npos42:   rmmovl %esi, 4(%ecx)    # ...and store it to dst

        mrmovl 8(%ebx), %esi    # read val from src...
        andl %esi, %esi             # val <= 0?
        jle Npos43             # if so, goto Npos:
        iaddl    $1, %eax           # count++
Npos43:   rmmovl %esi, 8(%ecx)    # ...and store it to dst

        mrmovl 12(%ebx), %esi  # read val from src...
        andl %esi, %esi             # val <= 0?
        jle Npos44             # if so, goto Npos:
        iaddl    $1, %eax           # count++
Npos44:   rmmovl %esi, 12(%ecx)   # ...and store it to dst

        mrmovl 16(%ebx), %esi  # read val from src...
```

```
	andl %esi, %esi		# val <= 0?
	jle Npos45		# if so, goto Npos:
	iaddl	$1, %eax	# count++
Npos45:	rmmovl %esi, 16(%ecx)	# ...and store it to dst

	mrmovl 20(%ebx), %esi	# read val from src...
	andl %esi, %esi		# val <= 0?
	jle Npos46		# if so, goto Npos:
	iaddl	$1, %eax	# count++
Npos46:	rmmovl %esi, 20(%ecx)	# ...and store it to dst

	mrmovl 24(%ebx), %esi	# read val from src...
	andl %esi, %esi		# val <= 0?
	jle Npos47		# if so, goto Npos:
	iaddl	$1, %eax	# count++
Npos47:	rmmovl %esi, 24(%ecx)	# ...and store it to dst

	mrmovl 28(%ebx), %esi	# read val from src...
	iaddl	$32,%ebx	# src+=8
	rmmovl %esi, 28(%ecx)	# ...and store it to dst
	andl %esi, %esi		# val <= 0?
	jle Npos48		# if so, goto Npos:
	iaddl	$1, %eax	# count++
Npos48:	iaddl	$32,%ecx	# dst+=8
	iaddl	$-8,%edx
######### End of 8-element checking and 'ncopy ##########


### Check if left elements are more than 4, and if so, ########
### we 'ncopy' 4 elements.		################
Loop3:	iaddl	$4,%edx		# len-=4 > 0 ?
	jl Loop2	# if so, goto Loop:

	mrmovl (%ebx), %esi	# read val from src...
	andl %esi, %esi		# val <= 0?
	jle Npos31		# if so, goto Npos:
	iaddl	$1, %eax	# count++
Npos31:	rmmovl %esi, (%ecx)	# ...and store it to dst

	mrmovl 4(%ebx), %esi	# read val from src...
	andl %esi, %esi		# val <= 0?
	jle Npos32		# if so, goto Npos:
	iaddl	$1, %eax	# count++
Npos32:	rmmovl %esi, 4(%ecx)	# ...and store it to dst
```

```
        mrmovl 8(%ebx), %esi    # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos33              # if so, goto Npos:
        iaddl   $1, %eax        # count++
Npos33:  rmmovl %esi, 8(%ecx)   # ...and store it to dst

        mrmovl 12(%ebx), %esi   # read val from src...
        iaddl   $16,%ebx        # src++++++++
        rmmovl %esi, 12(%ecx)   # ...and store it to dst
        andl %esi, %esi         # val <= 0?
        jle Npos34              # if so, goto Npos:
        iaddl   $1, %eax        # count++
Npos34:  iaddl   $16,%ecx       # dst++++++++
        iaddl   $-4,%edx
######### End of 4-element checking and 'ncopy ##########


### Check if left elements are more than 2, and if so, ########
### we 'ncopy' 2 elements.                ################
Loop2:    iaddl   $2,%edx       # len+2 < 0 ?
        jl Loop1          # if so, goto Loop1:
        mrmovl (%ebx), %esi     # read val from src...
        andl %esi, %esi         # val <= 0?
        jle Npos21              # if so, goto Npos:
        iaddl   $1, %eax        # count++
Npos21:  rmmovl %esi, (%ecx)    # ...and store it to dst
        mrmovl 4(%ebx), %esi    # read val from src...
        iaddl   $8,%ebx         # src++++
        rmmovl %esi, 4(%ecx)    # ...and store it to dst
        andl %esi, %esi         # val <= 0?
        jle Npos22              # if so, goto Npos:
        iaddl   $1, %eax        # count++
Npos22:  iaddl   $8, %ecx       # dst++++
        iaddl   $-2,%edx
######### End of 2-element checking and 'ncopy ##########


### Check if left elements are more than 1, and if so, ########
### we 'ncopy' 1 elements.                ################
Loop1:    iaddl $1,%edx         # len+1 < 0?
        jl Done                 # if so, goto Done:
        mrmovl (%ebx), %esi     # read val from src...
        iaddl   $4,%ebx         # src++
```

```
        rmmovl %esi, (%ecx)     # ...and store it to dst
        andl %esi, %esi         # val <= 0?
        jle Done        # if so, goto Npos:
        iaddl   $1, %eax        # count++
######### End of 1-element checking and 'ncopy ##########

# Do not modify the following section of code
# Function epilogue.
Done:
        popl %edi                       # Restore callee-save registers
        popl %ebx
        popl %esi
        rrmovl %ebp, %esp
        popl %ebp
        ret
##################################################################
# Keep the following label at the end of your function
End:
#/* $end ncopy-ys */
```