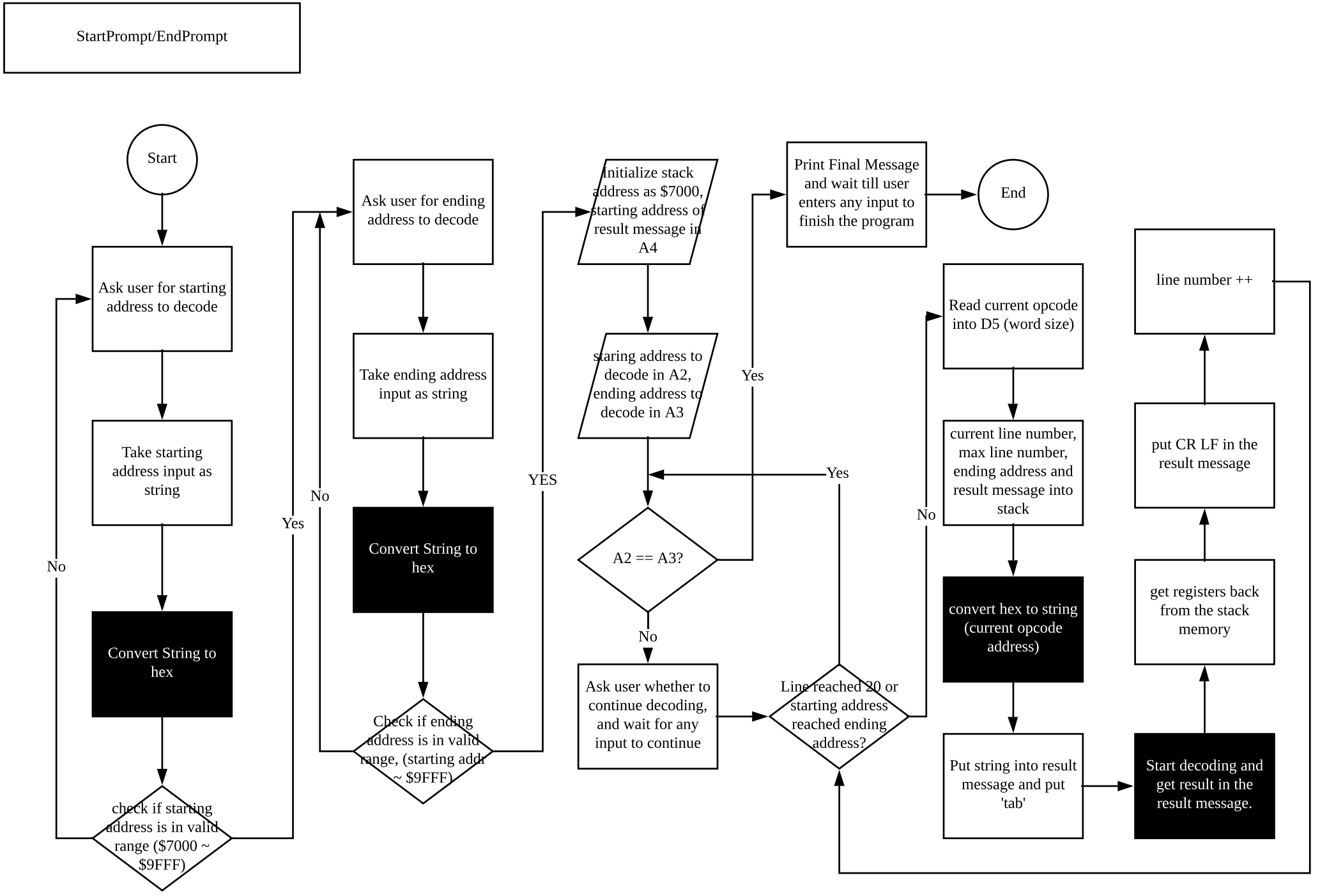
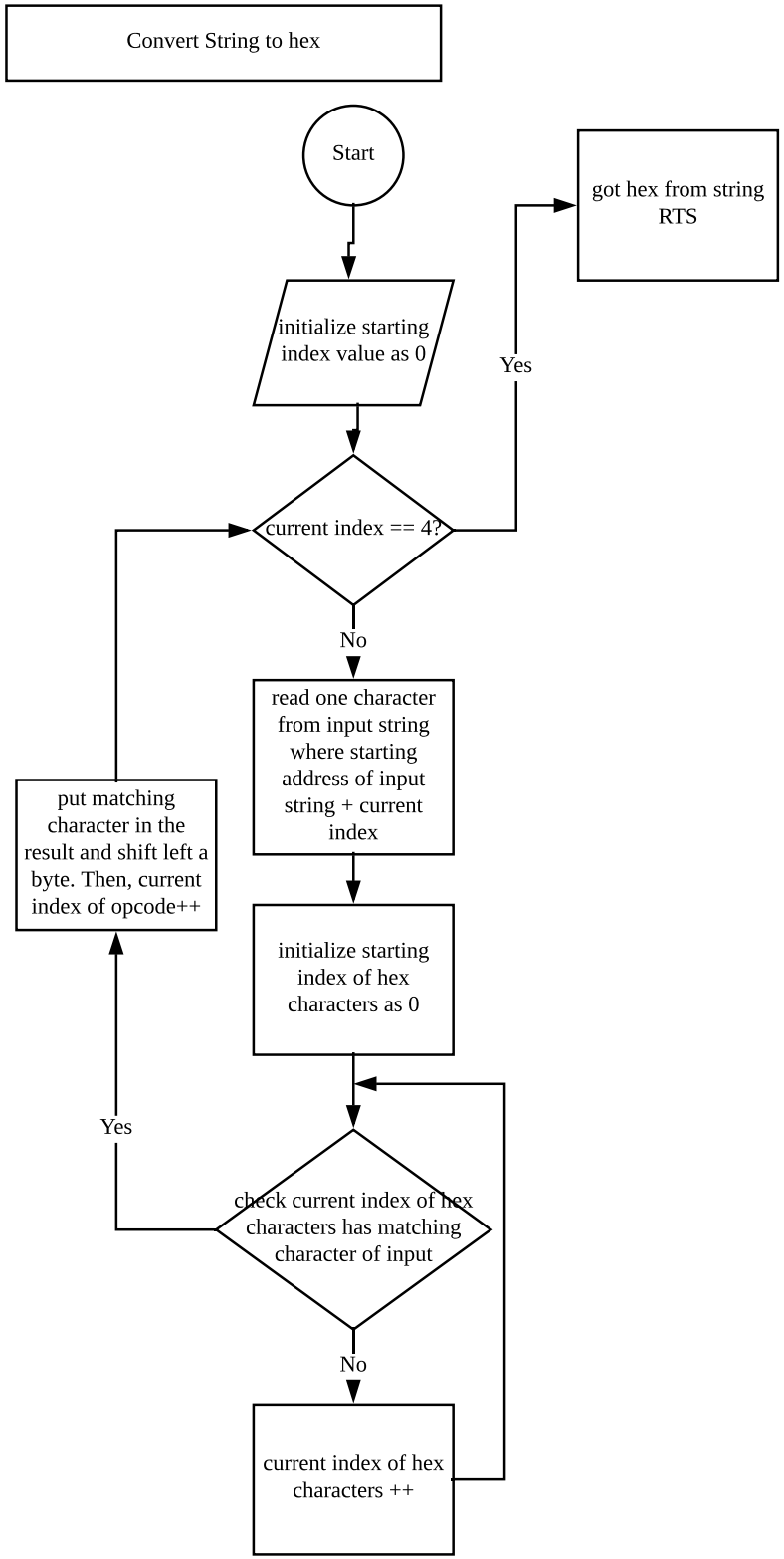
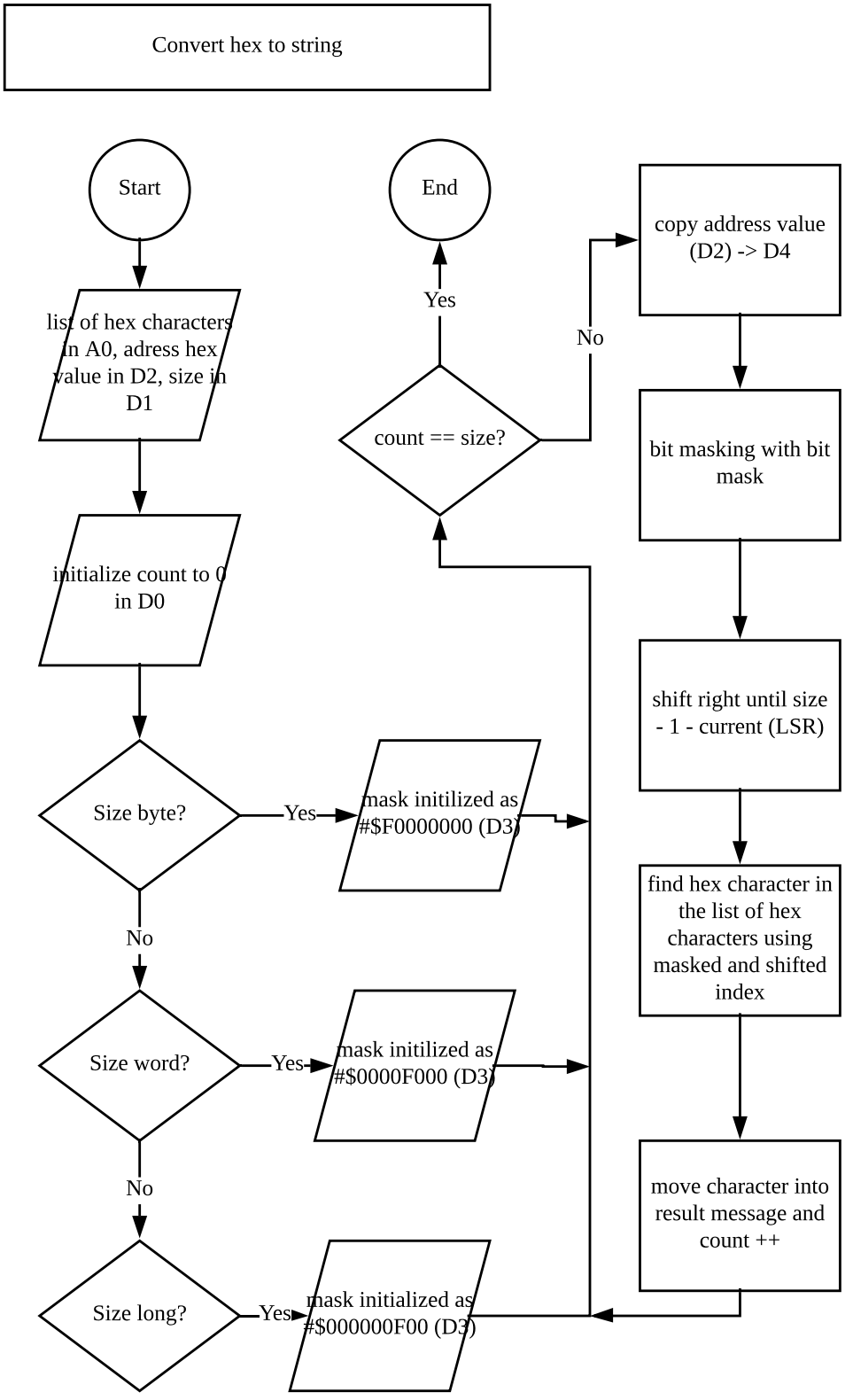

StartPrompt/EndPrompt
Start
Ask user for starting address to decode
Take starting address input as string
Convert String to hex
check if starting address is in valid range ($7000 ~ $9FFF)
No
Yes
Ask user for ending address to decode
Take ending address input as string
Convert String to hex
Check if ending address is in valid range, (starting addr ~ $9FFF)
No
YES
Initialize stack address as $7000, starting address of result message in A4
staring address to decode in A2, ending address to decode in A3
A2 == A3?
Yes
No
Ask user whether to continue decoding, and wait for any input to continue
Line reached 20 or starting address reached ending address?
Yes
No
Print Final Message and wait till user enters any input to finish the program
End
Read current opcode into D5 (word size)
current line number, max line number, ending address and result message into stack
convert hex to string (current opcode address)
Put string into result message and put 'tab'
Start decoding and get result in the result message.
get registers back from the stack memory
put CR LF in the result message
line number ++


Convert String to hex
Start
initialize starting index value as 0
current index == 4?
Yes
got hex from string RTS
No
read one character from input string where starting address of input string + current index
initialize starting index of hex characters as 0
check current index of hex characters has matching character of input
Yes
put matching character in the result and shift left a byte. Then, current index of opcode++
No
current index of hex characters ++


Convert hex to string
Start
list of hex characters in A0, adress hex value in D2, size in D1
initialize count to 0 in D0
Size byte?
Yes
mask initilized as #$F0000000 (D3)
No
Size word?
Yes
mask initilized as #$0000F000 (D3)
No
Size long?
Yes
mask initialized as #$000000F00 (D3)
count == size?
Yes
End
No
copy address value (D2) -> D4
bit masking with bit mask
shift right until size - 1 - current (LSR)
find hex character in the list of hex characters using masked and shifted index
move character into result message and count ++

Decoding

Start

Opcode == $4E75?

Yes

Put RTS in the result message and return to sub routine

No

Opcode start with $0?

Yes

Opcode starting with $00?

Yes

ORI_function

No

Opcode starting with $08?

Yes

BCLR_Immediate_EA function

No

Opcode starting with $0C

Yes

CMPI_function

No

BCLR_Dn_EA function

No

Opcode start with $1?

Yes

MOVE.B function

No

Opcode start with $2?

Yes

MOVE_MOVEA_L

No

Opcode start with $3?

Yes

MOVE_MOVEA_W

No

End

Opcode start with $4?

Yes

Opcode start with $44?

Yes

NEG_function

No

Opcode start with $48?

Yes

MOVEM_W_L_R to M

No

Opcode start with $4C?

Yes

MULS_DIVS_L_ MOVEM_W_L M to R

No

Opcode start with $4E?

Yes

JSR_function

No

LEA_function

No

Opcode start with $5?

Yes

SUBQ_function

End

No

Opcode start with $6?

Yes

Opcode start with $60?

Yes

BRA_function

No

Opcode start with $64?

Yes

BCC_function

No

Opcode start with $65?

Yes

BCS_function

No

Opcode start with $68?

Yes

BVC_function

No

Opcode start with $6C?

Yes

BGE_function

No

Opcode start with $6D?

Yes

BLT_function

No

Invalid_code

No

Opcode start with $8?

Yes

OR_DIVS_W

No

Opcode start with $9?

Yes

SUB_function

No

Opcode start with $B?

Yes

EOR_CMP

No

End

Opcode start with $C?

Yes

MULS_W_function

No

Opcode start with $D?

Yes

ADD_ADDA

No

Opcode start with $E?

Yes

AS_LS_RO_LR_BWL

ORI_function

Start

Save result message pointer (A5) to the stack

starting address of possible message in to A0 and A5

Put 'ORI.' in the A0 while incrementing A5 (end address) by one byte

Size bit == $00

Yes

put 'B\t#$' in the temp message while incrementing A5 by one byte

No

Size bit == $01

Yes

put 'W\t#$' in the temp message while incrementing A5 by one byte

No

Size bit == $02

Yes

put 'L\t#$' in the temp message while incrementing A5 by one byte

No

get result message pointer from the stack

Invalid_code

get next word size opcode from A2

get next long size opcode from Address A2

Check if first 4bits == $0

No

Yes

get next word size opcode from starting address A2

Convert hex immediate hex value to string and put in the temp message

get mode bits into D1, register bits in D0

put 'COMMA & SPACE' in the temp message

mode == $00

Yes

put 'D' in the temp message

put register number

No

mode == $02

Yes

put '(A' in the temp message

put register number

put ')+' in the temp message

No

mode == $03

Yes

put '-(A' in the temp message

put register number

put ')' in the temp message

No

mode == $04

Yes

put '(A' in the temp message

put register number

put ')' in the temp message

size == $00

Yes

read next opcode (word size)

put '$' in the temp message

No

size == $01

Yes

read next opcode (long size)

Convert hex to string (result in the temp message)

No

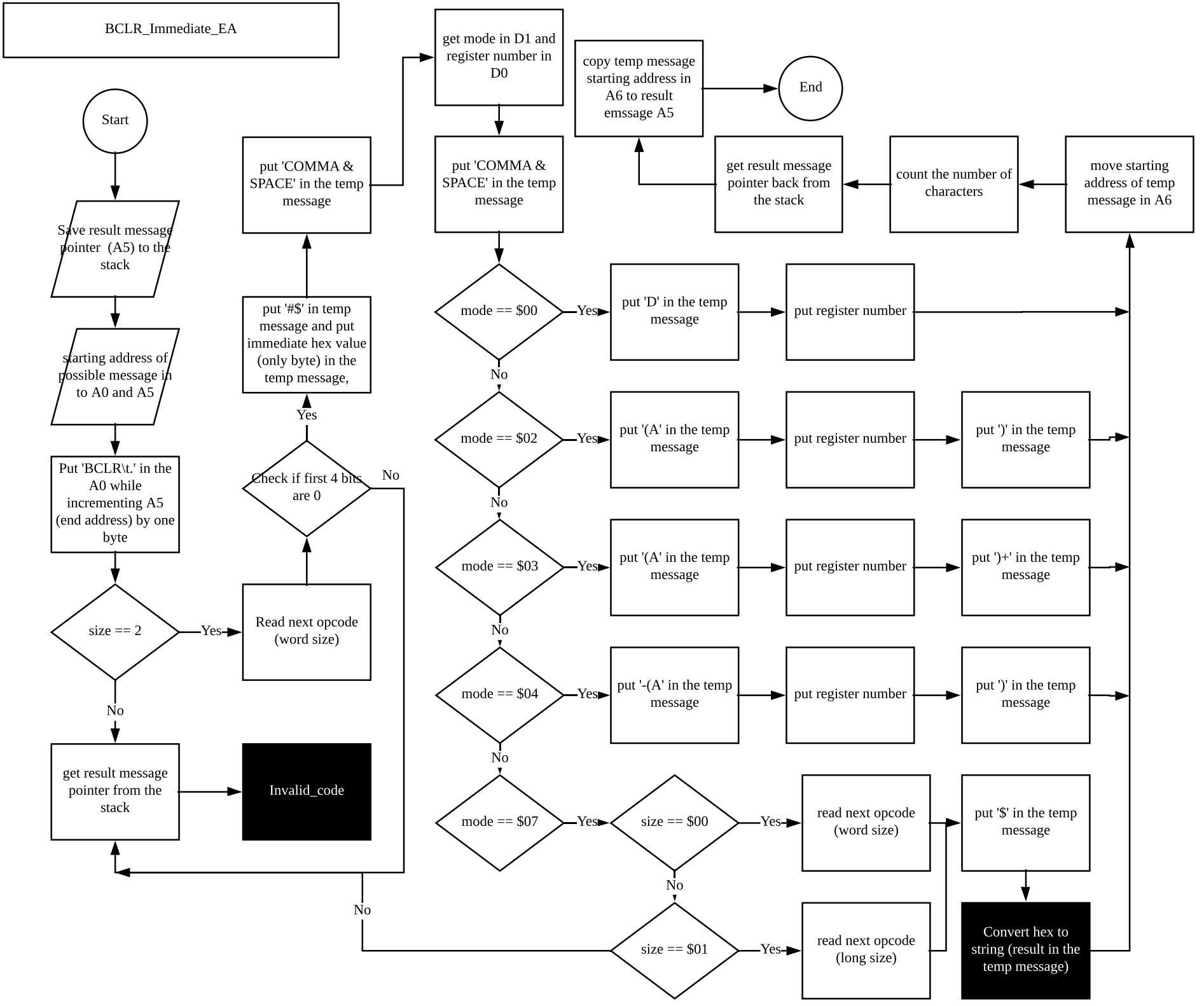
move starting address of temp message in A6

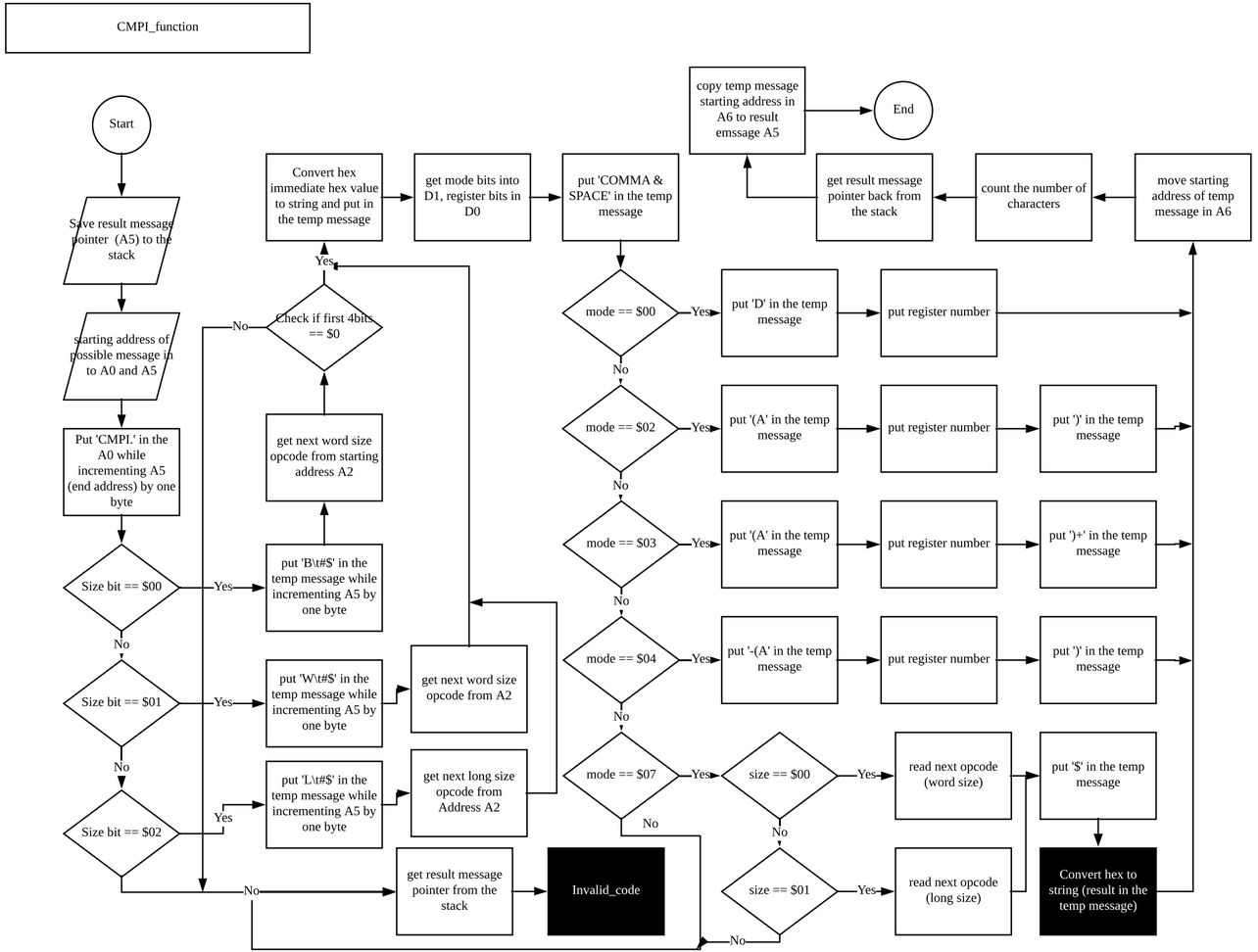
count the number of characters

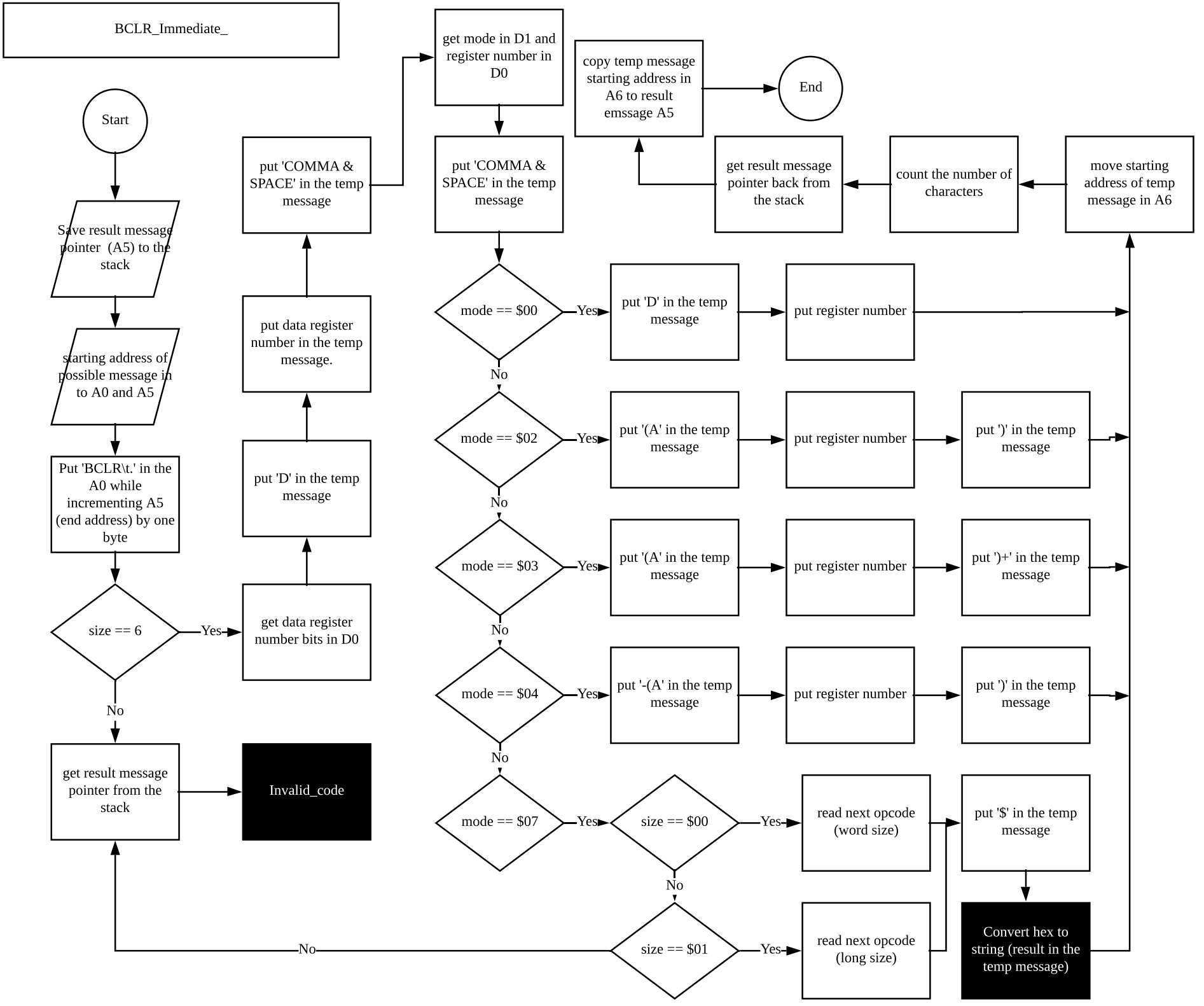
get result message pointer back from the stack

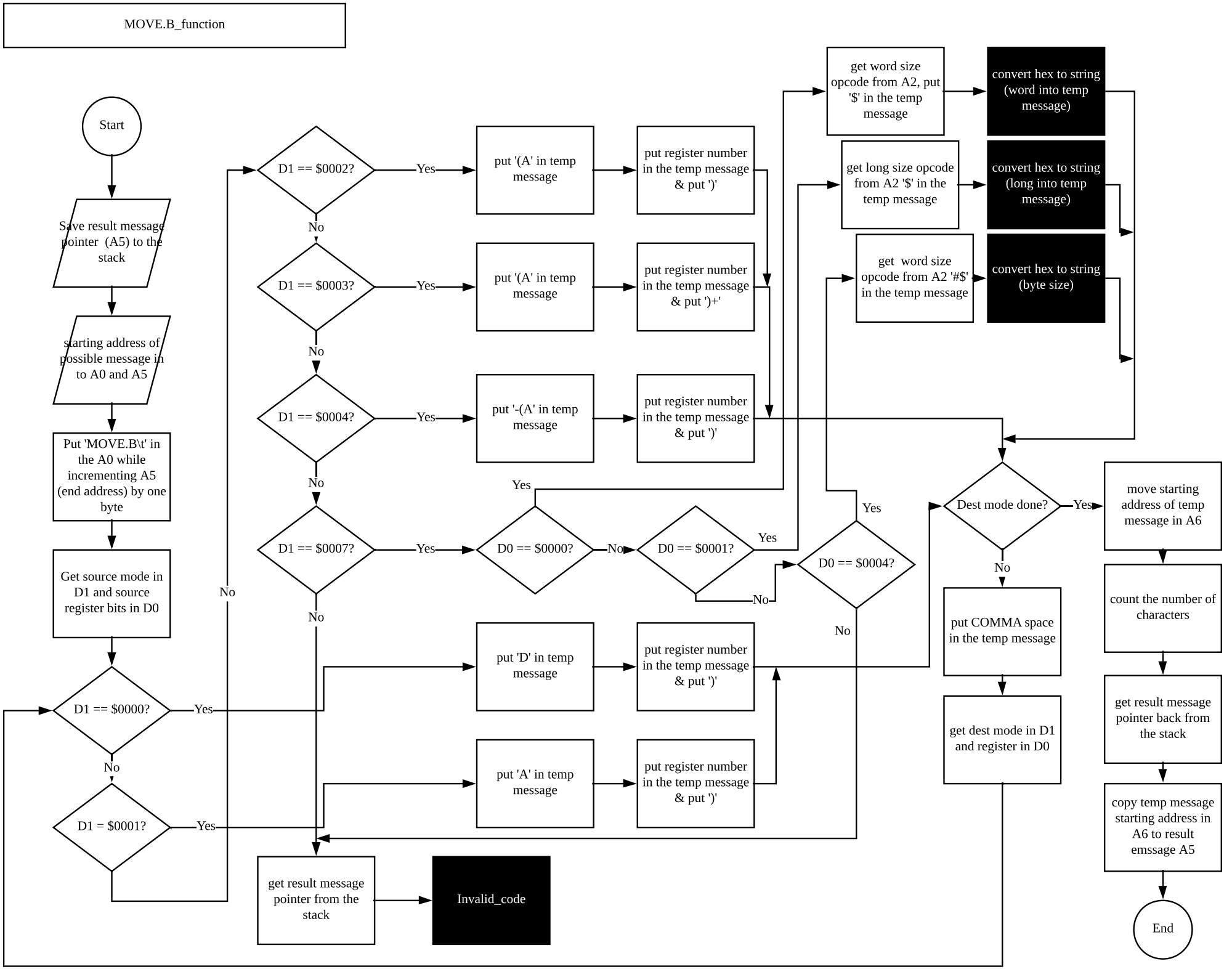
copy temp message starting address in A6 to result message A5

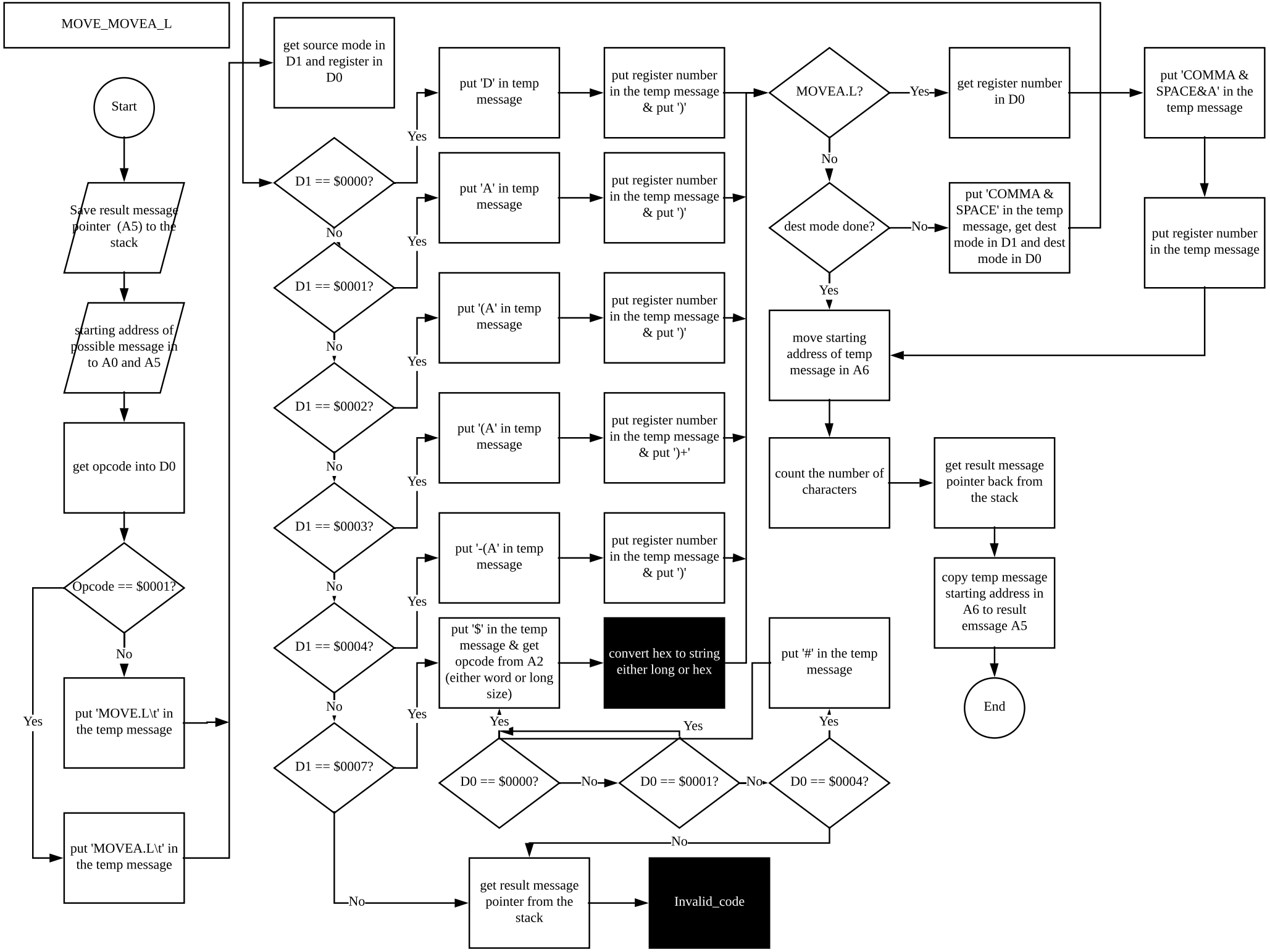
End

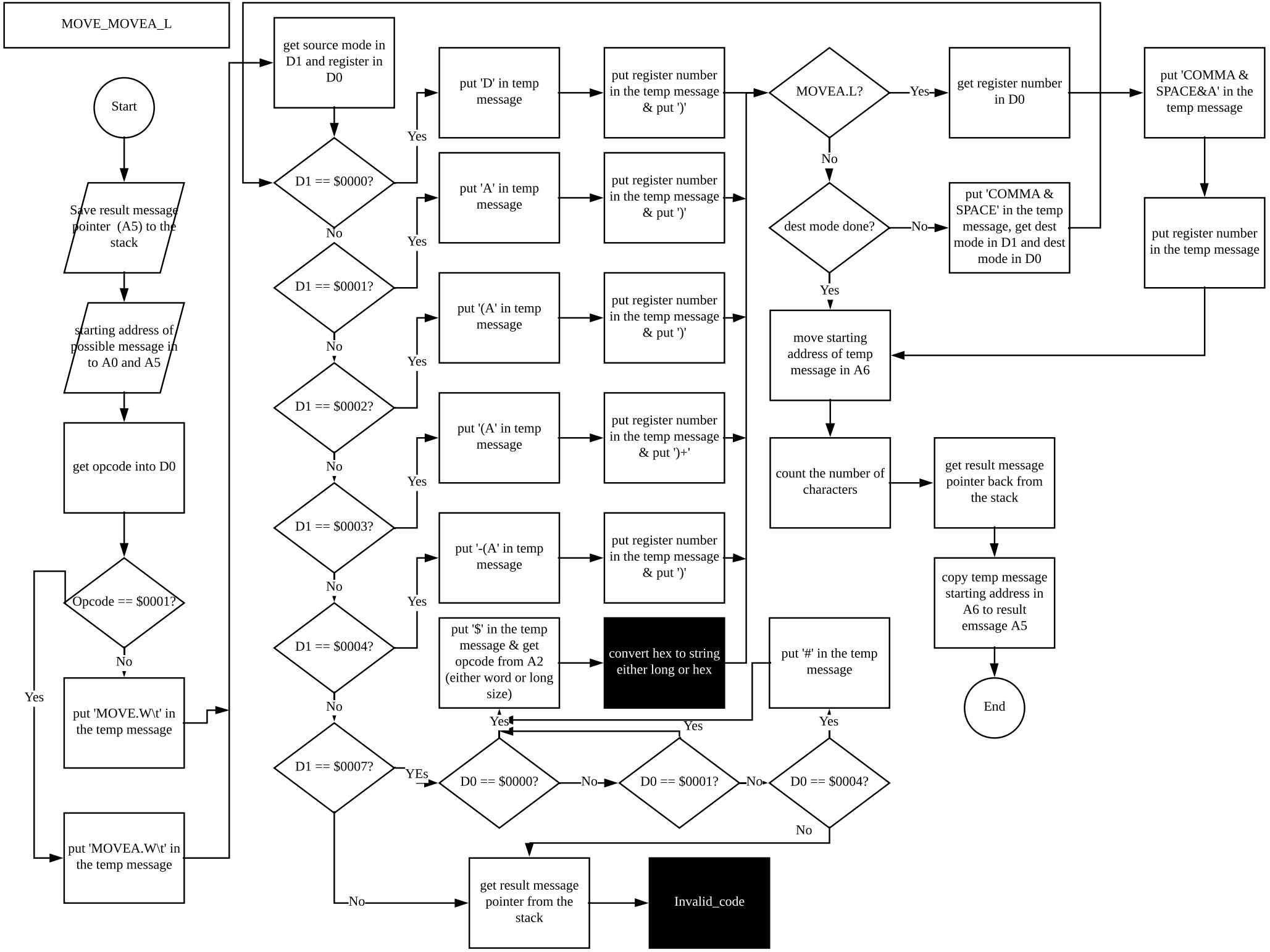

BCLR_Immediate_EA
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'BCLR\t.' in the A0 while incrementing A5 (end address) by one byte
size == 2
Yes
No
Read next opcode (word size)
Check if first 4 bits are 0
Yes
No
put '#$' in temp message and put immediate hex value (only byte) in the temp message,
put 'COMMA & SPACE' in the temp message
get mode in D1 and register number in D0
get result message pointer from the stack
Invalid_code
put 'COMMA & SPACE' in the temp message
mode == $00
Yes
put 'D' in the temp message
put register number
No
mode == $02
Yes
put '(A' in the temp message
put register number
put ')' in the temp message
No
mode == $03
Yes
put '(A' in the temp message
put register number
put ')+' in the temp message
No
mode == $04
Yes
put '-(A' in the temp message
put register number
put ')' in the temp message
No
mode == $07
Yes
size == $00
Yes
read next opcode (word size)
put '$' in the temp message
No
size == $01
Yes
read next opcode (long size)
Convert hex to string (result in the temp message)
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


CMPI_function
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'CMPI.' in the A0 while incrementing A5 (end address) by one byte
Size bit == $00
Yes
No
put 'B\t#$' in the temp message while incrementing A5 by one byte
Size bit == $01
Yes
No
put 'W\t#$' in the temp message while incrementing A5 by one byte
get next word size opcode from A2
Size bit == $02
Yes
No
put 'L\t#$' in the temp message while incrementing A5 by one byte
get next long size opcode from Address A2
get next word size opcode from starting address A2
Check if first 4bits == $0
Yes
No
Convert hex immediate hex value to string and put in the temp message
get mode bits into D1, register bits in D0
put 'COMMA & SPACE' in the temp message
mode == $00
Yes
No
put 'D' in the temp message
put register number
mode == $02
Yes
No
put '(A' in the temp message
put register number
put ')' in the temp message
mode == $03
Yes
No
put '(A' in the temp message
put register number
put ')+' in the temp message
mode == $04
Yes
No
put '-(A' in the temp message
put register number
put ')' in the temp message
mode == $07
Yes
No
size == $00
Yes
No
read next opcode (word size)
size == $01
Yes
No
read next opcode (long size)
put '$' in the temp message
Convert hex to string (result in the temp message)
get result message pointer from the stack
Invalid_code
get result message pointer back from the stack
count the number of characters
move starting address of temp message in A6
copy temp message starting address in A6 to result emssage A5
End


BCLR_Immediate_
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'BCLR\t.' in the A0 while incrementing A5 (end address) by one byte
size == 6
Yes
No
get data register number bits in D0
put data register number in the temp message.
put 'D' in the temp message
put 'COMMA & SPACE' in the temp message
get result message pointer from the stack
Invalid_code
get mode in D1 and register number in D0
put 'COMMA & SPACE' in the temp message
mode == $00
Yes
No
put 'D' in the temp message
put register number
mode == $02
put '(A' in the temp message
put register number
put ')' in the temp message
mode == $03
put '(A' in the temp message
put register number
put ')+' in the temp message
mode == $04
put '-(A' in the temp message
put register number
put ')' in the temp message
mode == $07
size == $00
read next opcode (word size)
put '$' in the temp message
size == $01
read next opcode (long size)
Convert hex to string (result in the temp message)
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


MOVE.B_function
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'MOVE.B\t' in the A0 while incrementing A5 (end address) by one byte
Get source mode in D1 and source register bits in D0
D1 == $0000?
Yes
No
D1 = $0001?
Yes
No
D1 == $0002?
Yes
No
D1 == $0003?
Yes
No
D1 == $0004?
Yes
No
D1 == $0007?
Yes
No
D0 == $0000?
No
D0 == $0001?
Yes
No
D0 == $0004?
Yes
No
put 'D' in temp message
put register number in the temp message & put ')'
put 'A' in temp message
put register number in the temp message & put ')'
put '(A' in temp message
put register number in the temp message & put ')'
put '(A' in temp message
put register number in the temp message & put ')+'
put '-(A' in temp message
put register number in the temp message & put ')'
get word size opcode from A2, put '$' in the temp message
convert hex to string (word into temp message)
get long size opcode from A2 '$' in the temp message
convert hex to string (long into temp message)
get word size opcode from A2 '#$' in the temp message
convert hex to string (byte size)
get result message pointer from the stack
Invalid_code
Dest mode done?
Yes
No
put COMMA space in the temp message
get dest mode in D1 and register in D0
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


MOVE_MOVEA_L
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
get opcode into D0
Opcode == $0001?
Yes
No
put 'MOVE.L\t' in the temp message
put 'MOVEA.L\t' in the temp message
get source mode in D1 and register in D0
D1 == $0000?
D1 == $0001?
D1 == $0002?
D1 == $0003?
D1 == $0004?
D1 == $0007?
put 'D' in temp message
put 'A' in temp message
put '(A' in temp message
put '(A' in temp message
put '-(A' in temp message
put '$' in the temp message & get opcode from A2 (either word or long size)
put register number in the temp message & put ')'
put register number in the temp message & put ')'
put register number in the temp message & put ')'
put register number in the temp message & put ')+'
put register number in the temp message & put ')'
convert hex to string either long or hex
D0 == $0000?
D0 == $0001?
D0 == $0004?
put '#' in the temp message
get result message pointer from the stack
Invalid_code
MOVEA.L?
get register number in D0
put 'COMMA & SPACE&A' in the temp message
dest mode done?
put 'COMMA & SPACE' in the temp message, get dest mode in D1 and dest mode in D0
put register number in the temp message
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


MOVE_MOVEA_L
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
get opcode into D0
Opcode == $0001?
Yes
No
put 'MOVE.W\t' in the temp message
put 'MOVEA.W\t' in the temp message
get source mode in D1 and register in D0
D1 == $0000?
D1 == $0001?
D1 == $0002?
D1 == $0003?
D1 == $0004?
D1 == $0007?
YEs
No
put 'D' in temp message
put 'A' in temp message
put '(A' in temp message
put '(A' in temp message
put '-(A' in temp message
put '$' in the temp message & get opcode from A2 (either word or long size)
put register number in the temp message & put ')'
put register number in the temp message & put ')'
put register number in the temp message & put ')'
put register number in the temp message & put ')+'
put register number in the temp message & put ')'
convert hex to string either long or hex
D0 == $0000?
D0 == $0001?
D0 == $0004?
put '#' in the temp message
get result message pointer from the stack
Invalid_code
MOVEA.L?
dest mode done?
get register number in D0
put 'COMMA & SPACE&A' in the temp message
put 'COMMA & SPACE' in the temp message, get dest mode in D1 and dest mode in D0
put register number in the temp message
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End

NEG

Start

Save result message pointer (A5) to the stack

starting address of possible message in to A0 and A5

Put 'NEG.' in the A0 while incrementing A5 (end address) by one byte

Size bit == $00

Yes

put 'B\t#$' in the temp message while incrementing A5 by one byte

No

Size bit == $01

Yes

put 'W\t#$' in the temp message while incrementing A5 by one byte

No

Size bit == $02

Yes

put 'L\t#$' in the temp message while incrementing A5 by one byte

No

get result message pointer from the stack

Invalid_code

get mode bits into D1, register bits in D0

put 'COMMA & SPACE' in the temp message

mode == $00

Yes

put 'D' in the temp message

put register number

No

mode == $02

Yes

put '(A' in the temp message

put register number

put ')' in the temp message

No

mode == $03

Yes

put '(A' in the temp message

put register number

put ')+' in the temp message

No

mode == $04

Yes

put '-(A' in the temp message

put register number

put ')' in the temp message

No

mode == $07

Yes

No

size == $00

Yes

read next opcode (word size)

put '$' in the temp message

No

size == $01

Yes

read next opcode (long size)

Convert hex to string (result in the temp message)

No

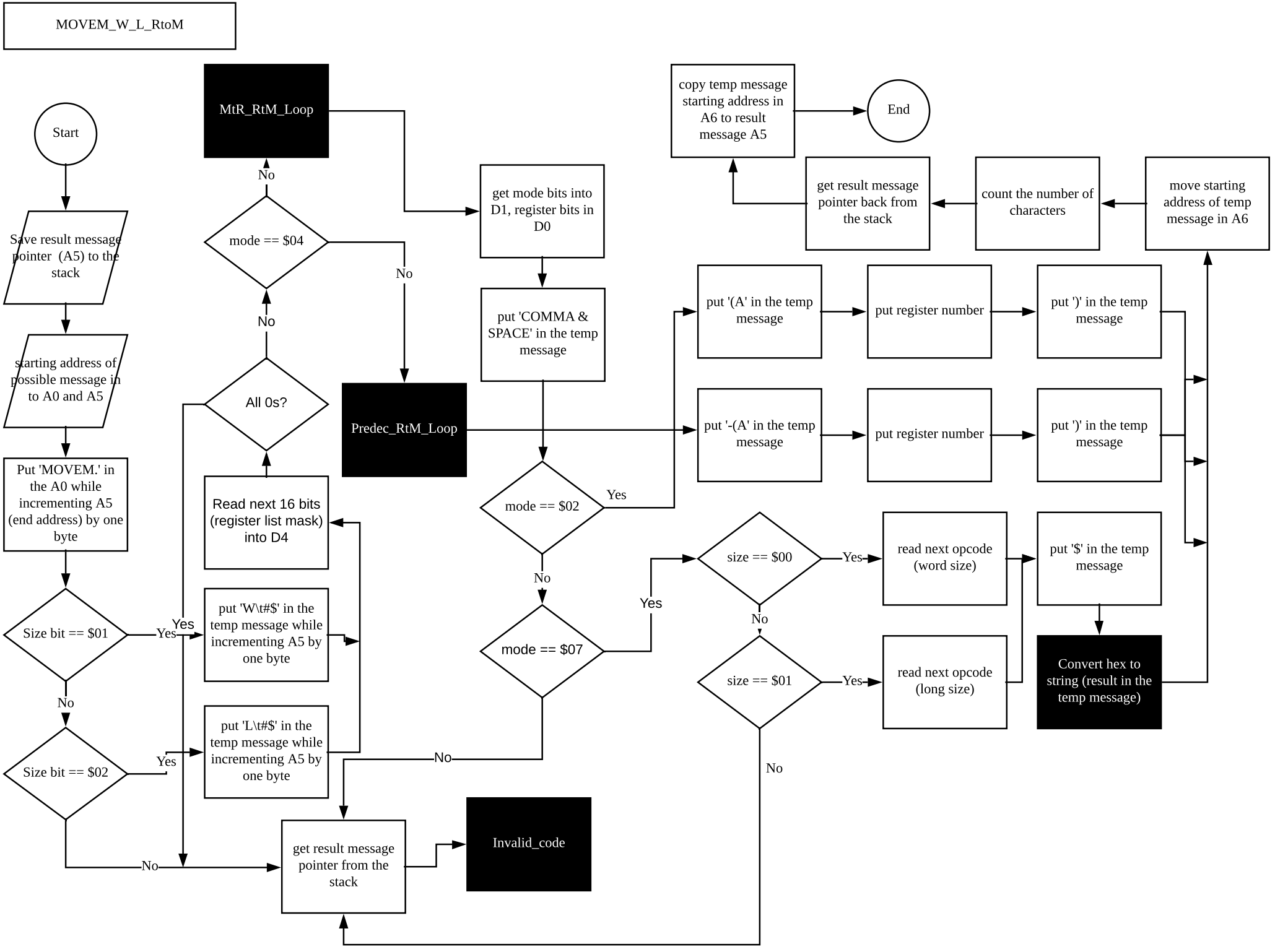
move starting address of temp message in A6

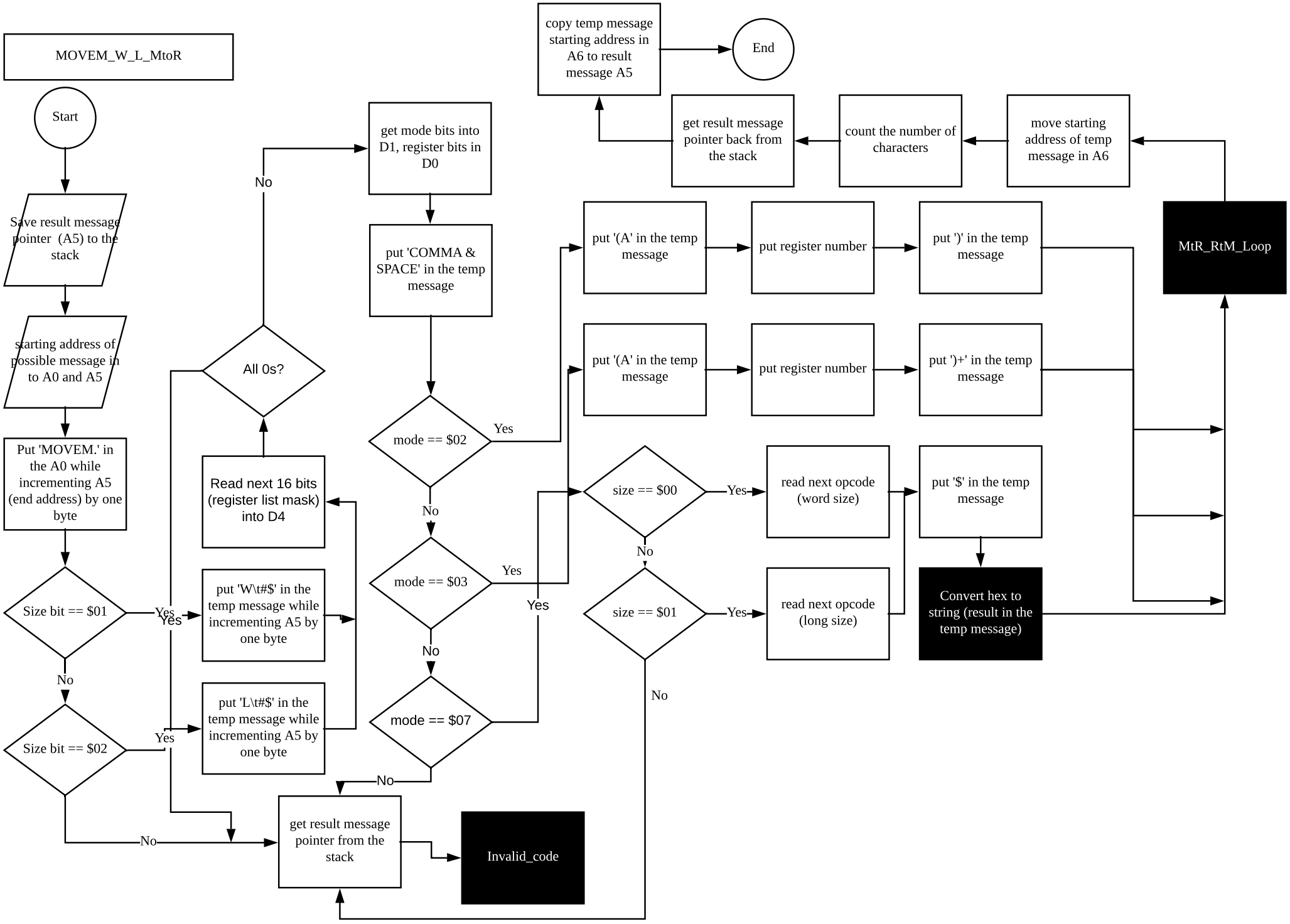
count the number of characters

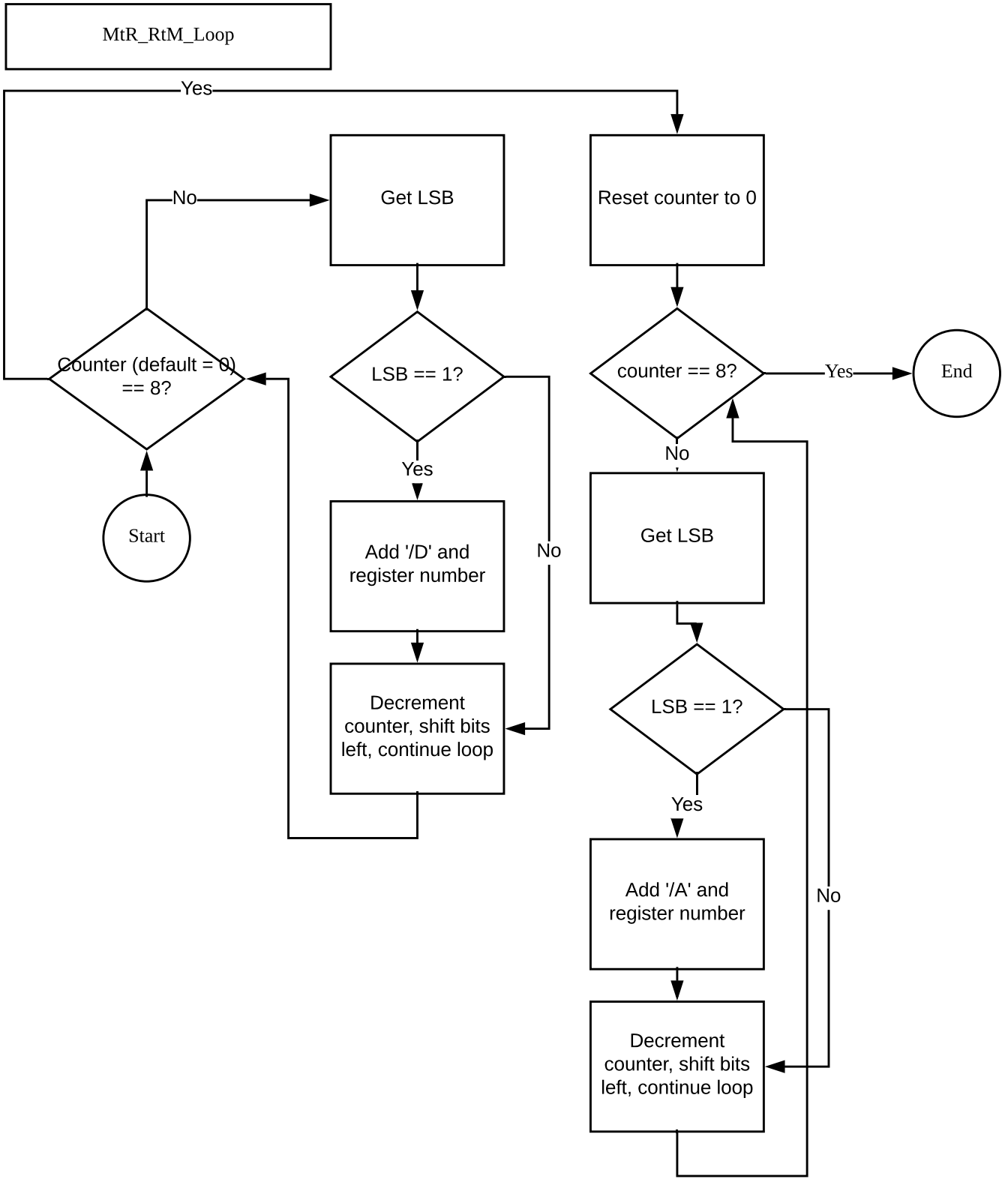
get result message pointer back from the stack

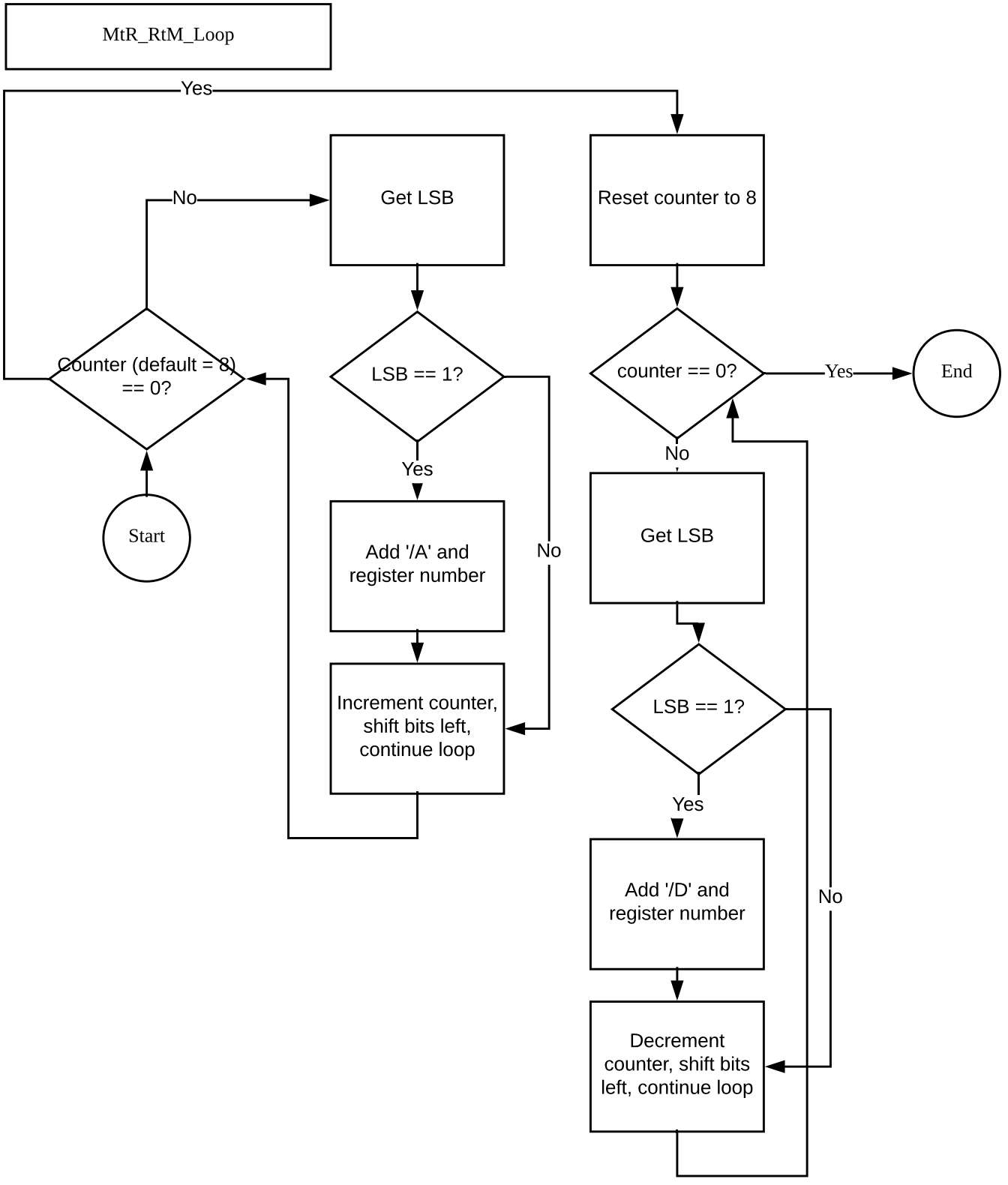
copy temp message starting address in A6 to result message A5

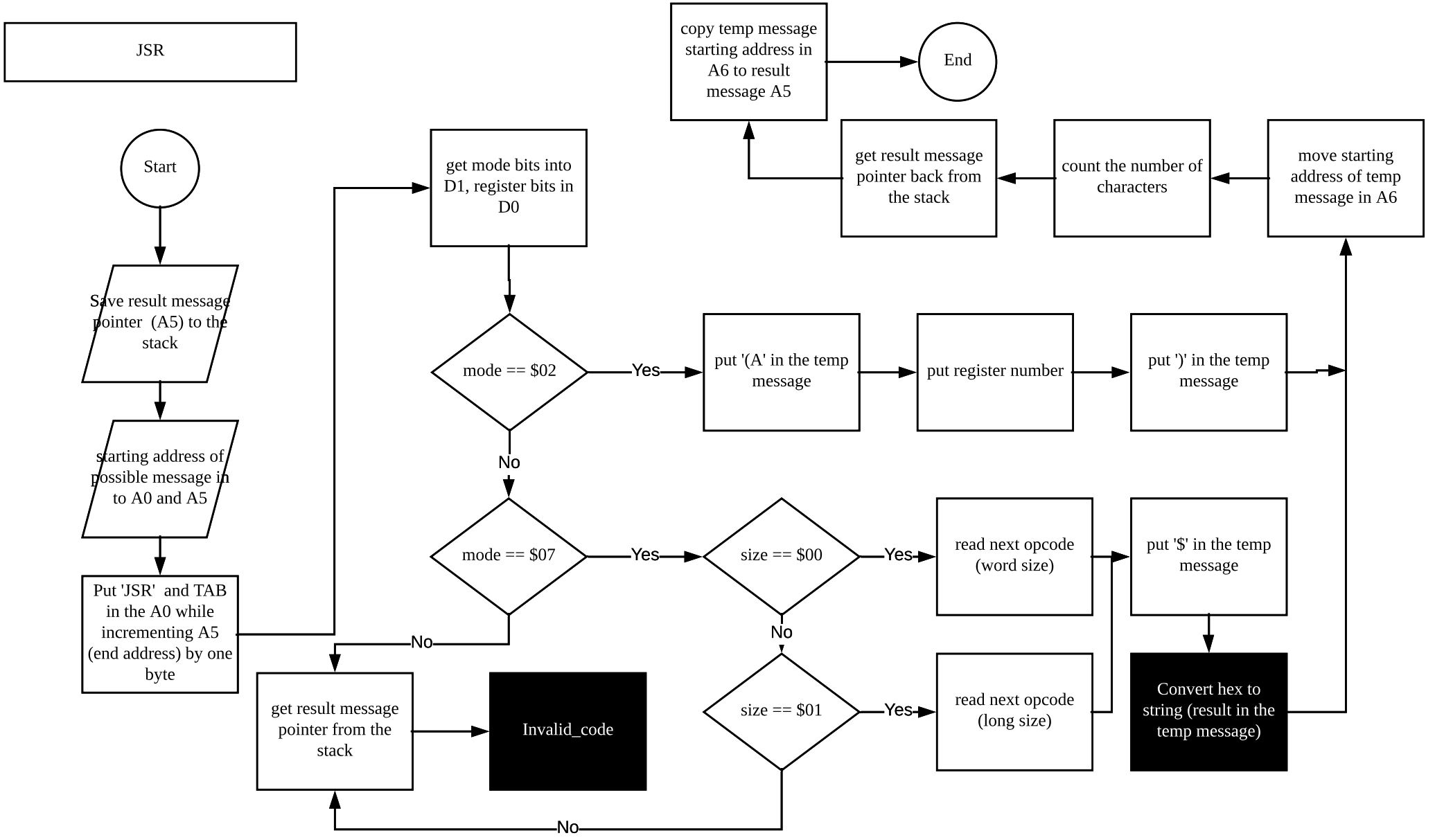
End

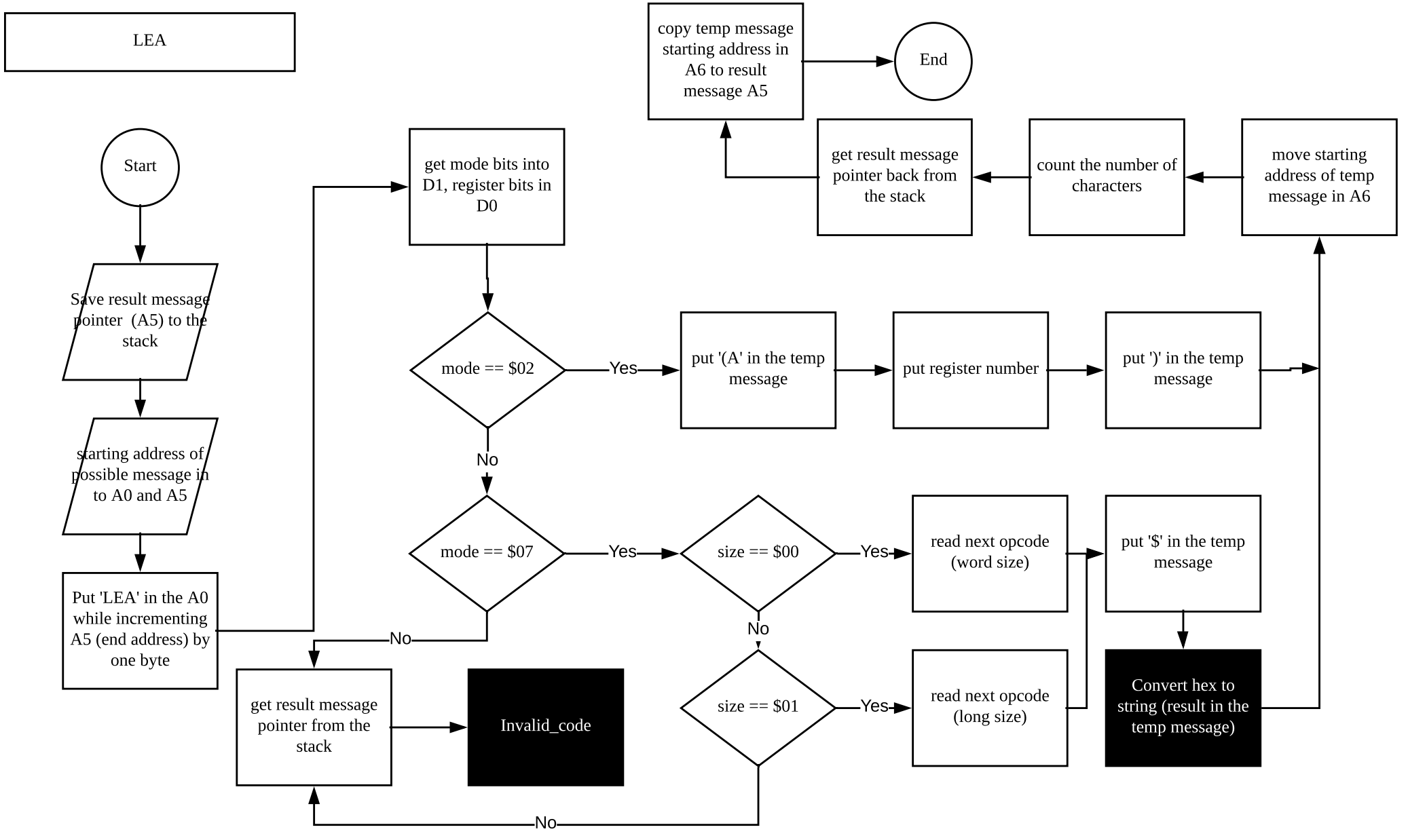

MOVEM_W_L_RtoM
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'MOVEM.' in the A0 while incrementing A5 (end address) by one byte
Size bit == $01
Yes
put 'W\t#$' in the temp message while incrementing A5 by one byte
No
Size bit == $02
Yes
put 'L\t#$' in the temp message while incrementing A5 by one byte
No
MtR_RtM_Loop
No
mode == $04
No
No
All 0s?
Read next 16 bits (register list mask) into D4
Predec_RtM_Loop
get mode bits into D1, register bits in D0
put 'COMMA & SPACE' in the temp message
mode == $02
Yes
No
mode == $07
Yes
No
get result message pointer from the stack
Invalid_code
put '(A' in the temp message
put register number
put ')' in the temp message
put '-(A' in the temp message
put register number
put ')' in the temp message
size == $00
Yes
read next opcode (word size)
No
size == $01
Yes
read next opcode (long size)
No
put '$' in the temp message
Convert hex to string (result in the temp message)
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result message A5
End

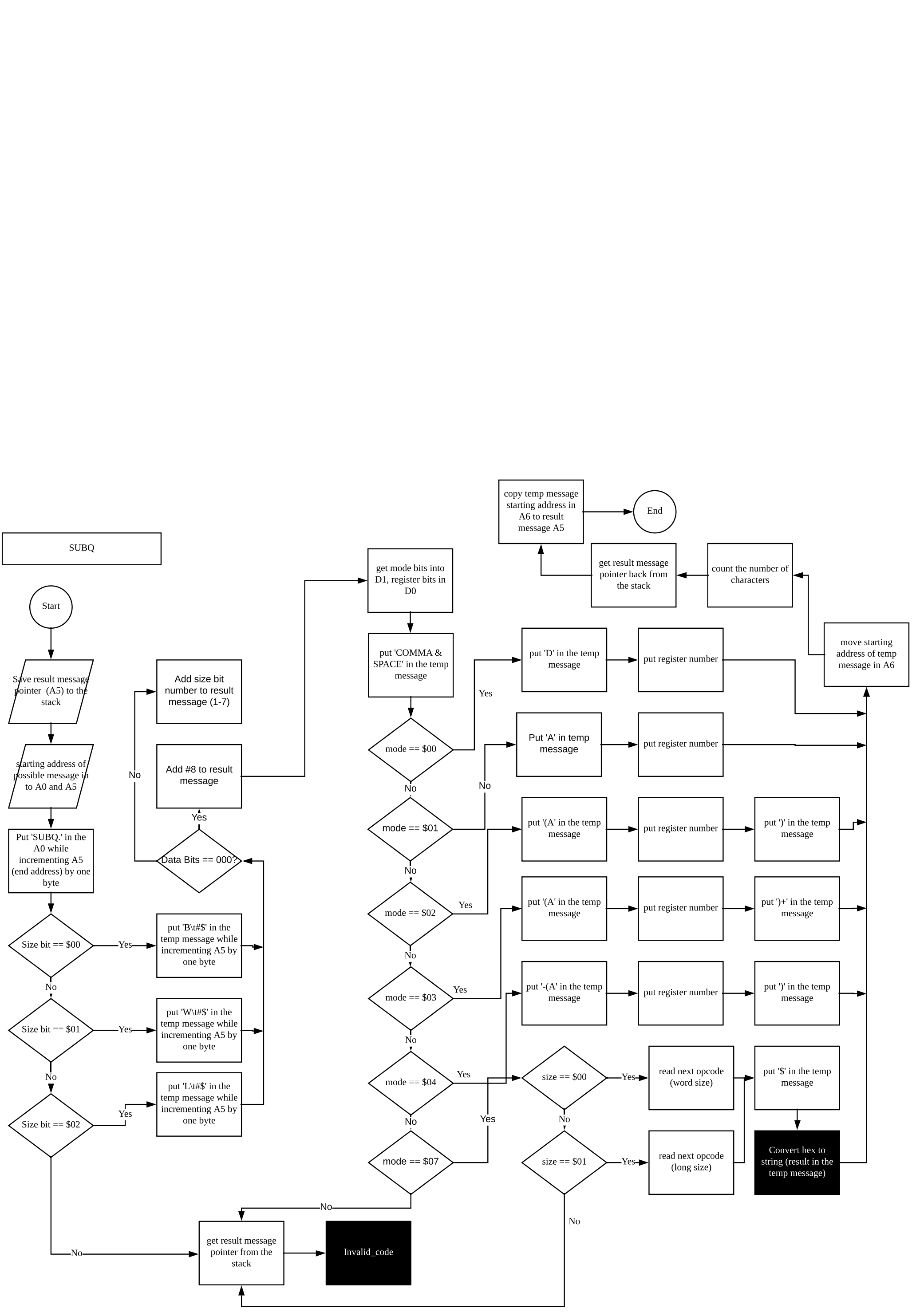

MOVEM_W_L_MtoR
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'MOVEM.' in the A0 while incrementing A5 (end address) by one byte
Size bit == $01
Yes
No
Size bit == $02
Yes
No
put 'W\t#$' in the temp message while incrementing A5 by one byte
put 'L\t#$' in the temp message while incrementing A5 by one byte
Read next 16 bits (register list mask) into D4
All 0s?
No
get mode bits into D1, register bits in D0
put 'COMMA & SPACE' in the temp message
mode == $02
Yes
No
mode == $03
Yes
No
mode == $07
No
get result message pointer from the stack
Invalid_code
put '(A' in the temp message
put register number
put ')' in the temp message
put '(A' in the temp message
put register number
put ')+' in the temp message
size == $00
Yes
No
size == $01
Yes
No
read next opcode (word size)
read next opcode (long size)
put '$' in the temp message
Convert hex to string (result in the temp message)
MtR_RtM_Loop
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result message A5
End

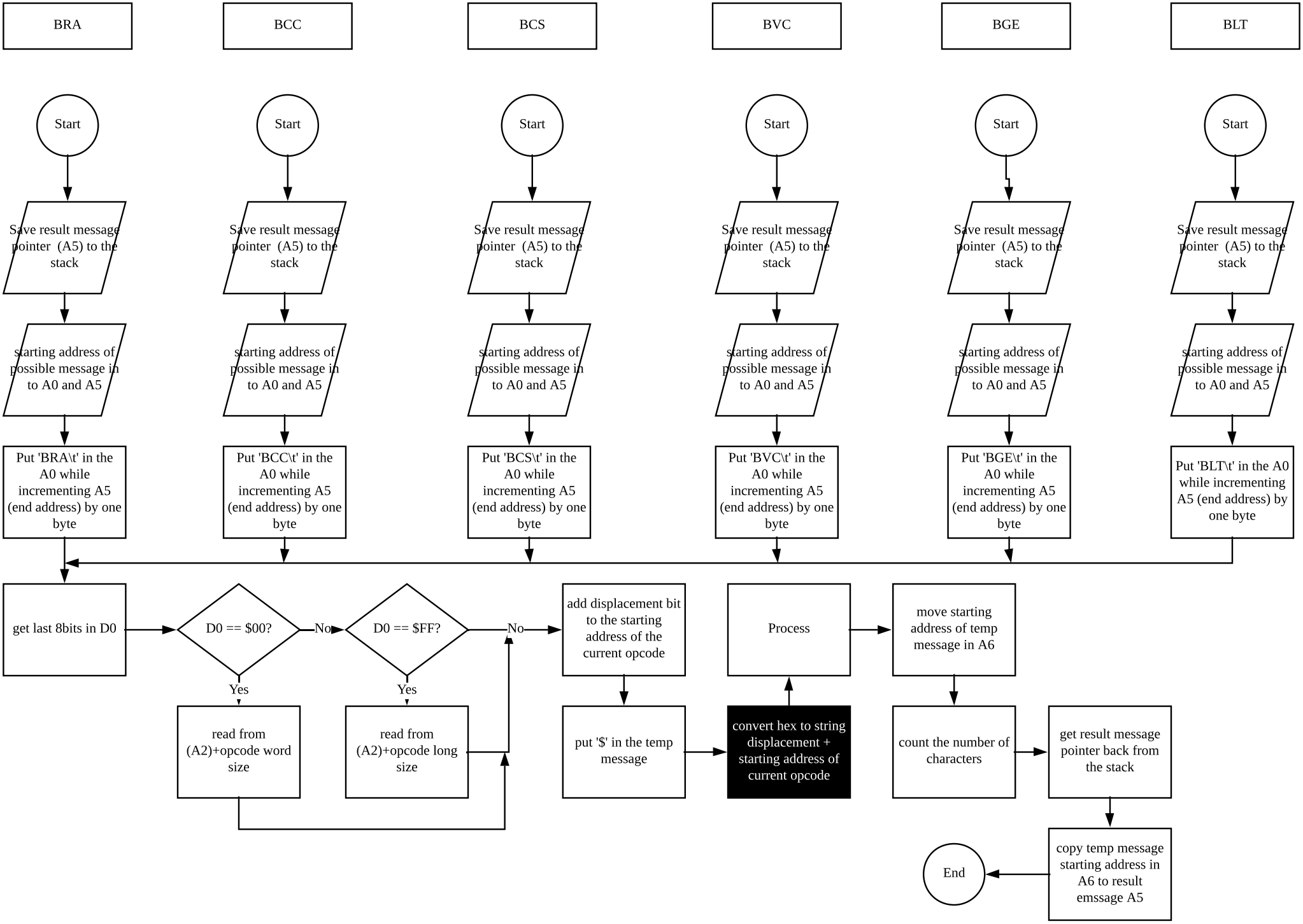

MtR_RtM_Loop
Yes
No
Get LSB
Reset counter to 0
Counter (default = 0) == 8?
LSB == 1?
counter == 8?
Yes
End
Start
Yes
No
Add '/D' and register number
Get LSB
Decrement counter, shift bits left, continue loop
LSB == 1?
Yes
Add '/A' and register number
No
Decrement counter, shift bits left, continue loop

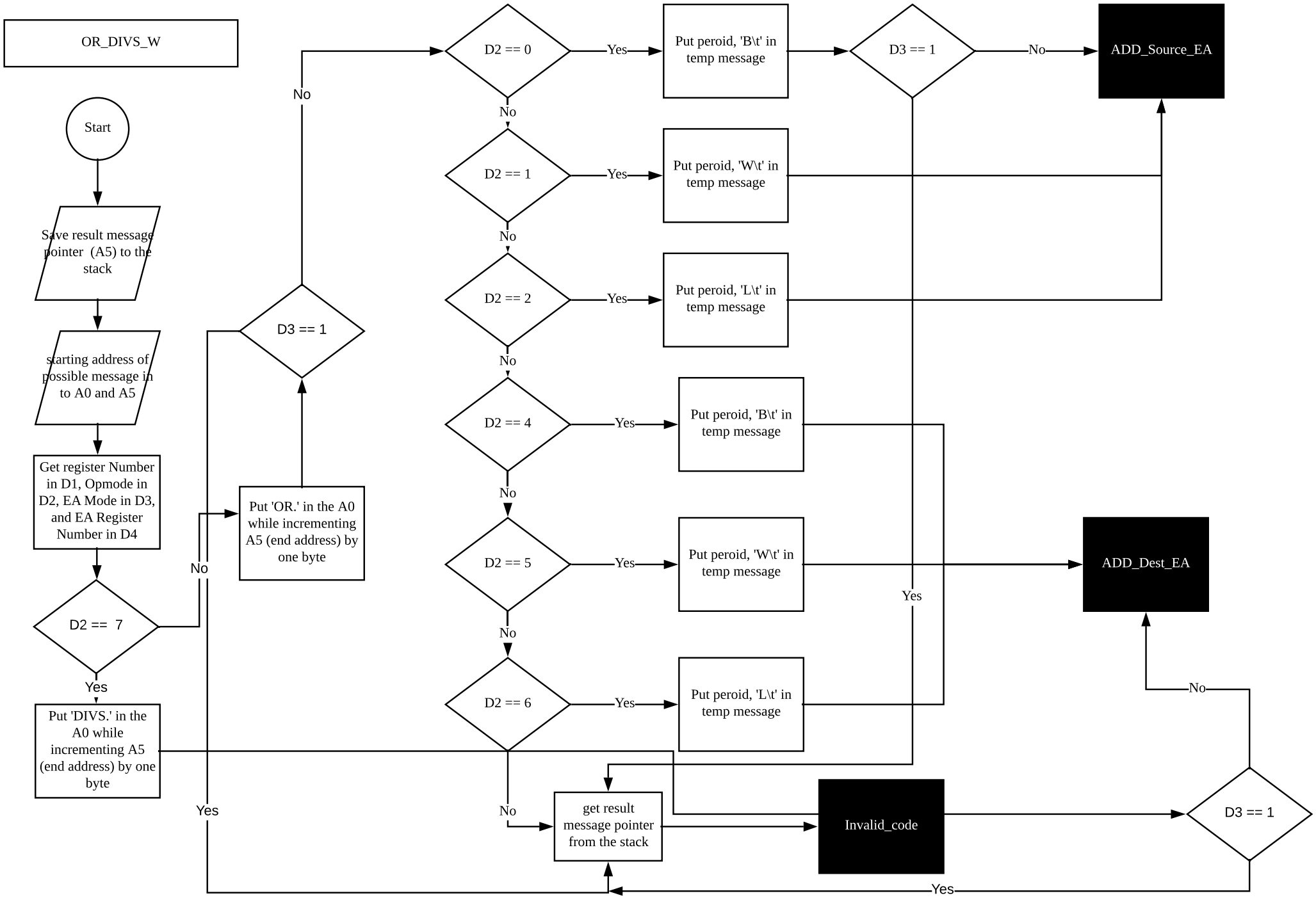

MtR_RtM_Loop
Yes
Reset counter to 8
Get LSB
No
Counter (default = 8) == 0?
LSB == 1?
counter == 0?
Yes
End
Start
Yes
No
Add '/A' and register number
Get LSB
Increment counter, shift bits left, continue loop
LSB == 1?
Yes
Add '/D' and register number
No
Decrement counter, shift bits left, continue loop

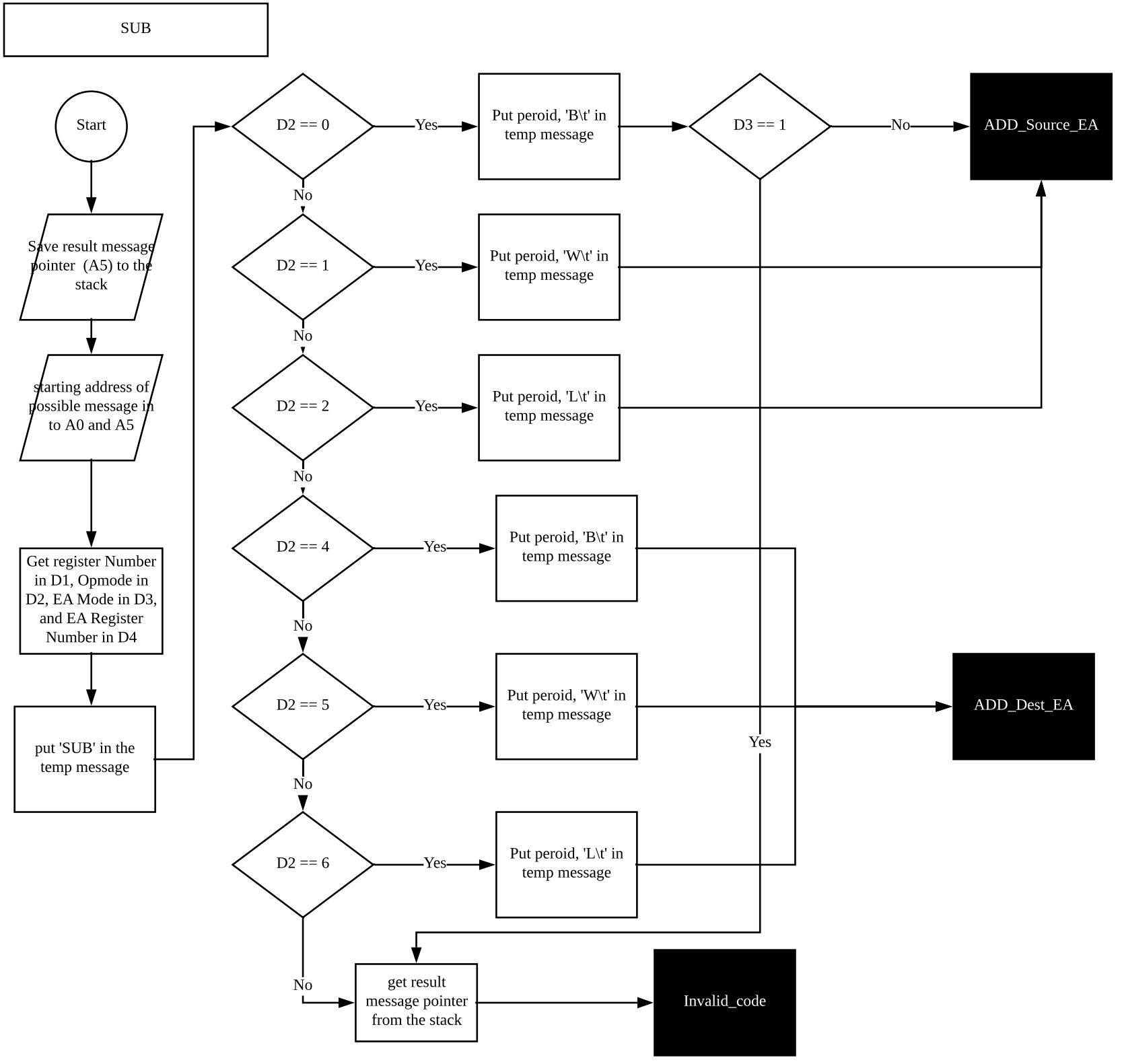

JSR
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'JSR' and TAB in the A0 while incrementing A5 (end address) by one byte
get mode bits into D1, register bits in D0
mode == $02
Yes
put '(A' in the temp message
put register number
put ')' in the temp message
No
mode == $07
Yes
size == $00
Yes
read next opcode (word size)
put '$' in the temp message
No
size == $01
Yes
read next opcode (long size)
Convert hex to string (result in the temp message)
No
get result message pointer from the stack
Invalid_code
copy temp message starting address in A6 to result message A5
End
get result message pointer back from the stack
count the number of characters
move starting address of temp message in A6

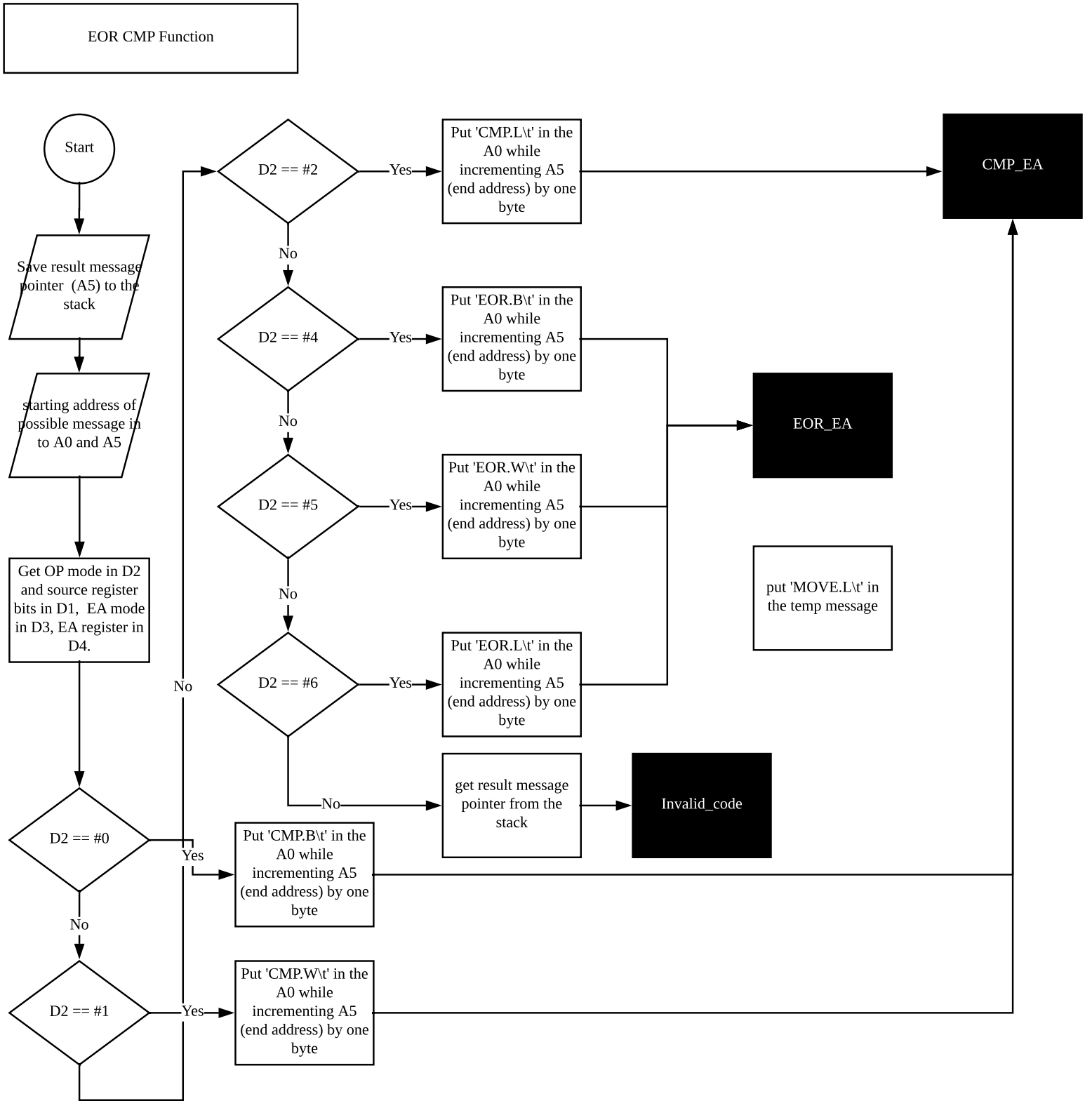

LEA
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'LEA' in the A0 while incrementing A5 (end address) by one byte
get mode bits into D1, register bits in D0
mode == $02
Yes
put '(A' in the temp message
put register number
put ')' in the temp message
No
mode == $07
Yes
size == $00
Yes
read next opcode (word size)
put '$' in the temp message
No
No
get result message pointer from the stack
Invalid_code
size == $01
Yes
read next opcode (long size)
Convert hex to string (result in the temp message)
No
copy temp message starting address in A6 to result message A5
End
get result message pointer back from the stack
count the number of characters
move starting address of temp message in A6


SUBQ
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'SUBQ.' in the A0 while incrementing A5 (end address) by one byte
Size bit == $00
Yes
put 'B\t#$' in the temp message while incrementing A5 by one byte
No
Size bit == $01
Yes
put 'W\t#$' in the temp message while incrementing A5 by one byte
No
Size bit == $02
Yes
put 'L\t#$' in the temp message while incrementing A5 by one byte
No
No
Add size bit number to result message (1-7)
Add #8 to result message
Yes
Data Bits == 000?
get mode bits into D1, register bits in D0
put 'COMMA & SPACE' in the temp message
mode == $00
Yes
put 'D' in the temp message
put register number
No
mode == $01
Yes
Put 'A' in temp message
put register number
No
mode == $02
Yes
put '(A' in the temp message
put register number
put ')' in the temp message
No
mode == $03
Yes
put '(A' in the temp message
put register number
put ')+' in the temp message
No
mode == $04
Yes
put '-(A' in the temp message
put register number
put ')' in the temp message
No
mode == $07
Yes
size == $00
Yes
read next opcode (word size)
put '$' in the temp message
No
size == $01
Yes
read next opcode (long size)
Convert hex to string (result in the temp message)
No
No
get result message pointer from the stack
Invalid_code
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result message A5
End


BRA
BCC
BCS
BVC
BGE
BLT
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Put 'BRA\t' in the A0 while incrementing A5 (end address) by one byte
Put 'BCC\t' in the A0 while incrementing A5 (end address) by one byte
Put 'BCS\t' in the A0 while incrementing A5 (end address) by one byte
Put 'BVC\t' in the A0 while incrementing A5 (end address) by one byte
Put 'BGE\t' in the A0 while incrementing A5 (end address) by one byte
Put 'BLT\t' in the A0 while incrementing A5 (end address) by one byte
get last 8bits in D0
D0 == $00?
No
D0 == $FF?
No
Yes
Yes
read from (A2)+opcode word size
read from (A2)+opcode long size
add displacement bit to the starting address of the current opcode
put '$' in the temp message
Process
convert hex to string displacement + starting address of current opcode
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


OR_DIVS_W
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Get register Number in D1, Opmode in D2, EA Mode in D3, and EA Register Number in D4
D2 == 7
No
Yes
Put 'DIVS.' in the A0 while incrementing A5 (end address) by one byte
Put 'OR.' in the A0 while incrementing A5 (end address) by one byte
D3 == 1
No
Yes
D2 == 0
Yes
Put peroid, 'B\t' in temp message
No
D2 == 1
Yes
Put peroid, 'W\t' in temp message
No
D2 == 2
Yes
Put peroid, 'L\t' in temp message
No
D2 == 4
Yes
Put peroid, 'B\t' in temp message
No
D2 == 5
Yes
Put peroid, 'W\t' in temp message
No
D2 == 6
Yes
Put peroid, 'L\t' in temp message
No
get result message pointer from the stack
D3 == 1
No
ADD_Source_EA
Yes
ADD_Dest_EA
Invalid_code
No
D3 == 1
Yes


SUB
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Get register Number in D1, Opmode in D2, EA Mode in D3, and EA Register Number in D4
put 'SUB' in the temp message
D2 == 0
Yes
Put peroid, 'B\t' in temp message
D3 == 1
No
ADD_Source_EA
No
D2 == 1
Yes
Put peroid, 'W\t' in temp message
No
D2 == 2
Yes
Put peroid, 'L\t' in temp message
No
D2 == 4
Yes
Put peroid, 'B\t' in temp message
No
D2 == 5
Yes
Put peroid, 'W\t' in temp message
ADD_Dest_EA
No
D2 == 6
Yes
Put peroid, 'L\t' in temp message
Yes
No
get result message pointer from the stack
Invalid_code


EOR CMP Function
Start
Save result message pointer (A5) to the stack
starting address of possible message in to A0 and A5
Get OP mode in D2 and source register bits in D1, EA mode in D3, EA register in D4.
D2 == #0
Yes
No
Put 'CMP.B\t' in the A0 while incrementing A5 (end address) by one byte
D2 == #1
Yes
Put 'CMP.W\t' in the A0 while incrementing A5 (end address) by one byte
No
D2 == #2
Yes
Put 'CMP.L\t' in the A0 while incrementing A5 (end address) by one byte
CMP_EA
No
D2 == #4
Yes
Put 'EOR.B\t' in the A0 while incrementing A5 (end address) by one byte
EOR_EA
No
D2 == #5
Yes
Put 'EOR.W\t' in the A0 while incrementing A5 (end address) by one byte
No
D2 == #6
Yes
Put 'EOR.L\t' in the A0 while incrementing A5 (end address) by one byte
No
get result message pointer from the stack
Invalid_code
put 'MOVE.L\t' in the temp message

MULS.W Function

Start

Save result message pointer (A5) to the stack

starting address of possible message in to A0 and A5

Get Op Mode in D2

D2 ==7

Yes

No

Invalid_code

Put 'MULS.W\t' in the A0 while incrementing A5 (end address) by one byte

Get Source register number in D1, EA Mode in D3, and EA Register Number in D4

D3 == 0

Yes

PUT 'D', EA register number, comma, space, 'D'', and Source register number.

No

D3 == 2

Yes

PUT '(A', EA register number, ')', comma, space, and Source register number.

No

D3 == 3

Yes

PUT '(A', EA register number, ')+', comma, space, and Source register number.

No

D3 == 4

Yes

PUT '-(A', EA register number, ')', comma, space, 'D'', and Source register number.

No

D3 == 7

Yes

D4 == 0

Yes

get word size opcode from A2, put '$' in the temp message

convert hex to string (word into temp message)

No

D4 == 1

Yes

get long size opcode from A2 '$' in the temp message

convert hex to string (long into temp message)

No

D4 == 4

Yes

put '#$' in the temp message while incrementing A5 by one byte

get next word size opcode from A2

Convert hex immediate hex value to string and put in the temp message

No

get result message pointer from the stack

'D', register number, Put, Comma, Space,

move starting address of temp message in A6

count the number of characters

get result message pointer back from the stack

copy temp message starting address in A6 to result emssage A5

End

ADD_ADDA Function

Start

Save result message pointer (A5) to the stack

starting address of possible message in to A0 and A5

Get register Number in D1, Op Code in D2, EA Mode in D3, and EA Register Number in D4

put 'ADD' in the temp message

D2 == 0

Yes

Put peroid, 'B\t' in temp message

D3 == 1

No

ADD_Source_EA

No

D2 == 1

Yes

Put peroid, 'W\t' in temp message

No

D2 == 2

Yes

Put peroid, 'L\t' in temp message

No

D2 == 4

Yes

Put peroid, 'B\t' in temp message

No

D2 == 5

Yes

Put peroid, 'W\t' in temp message

ADD_Dest_EA

No

D2 == 6

Yes

Put peroid, 'L\t' in temp message

Yes

No

D2 == 3

Yes

Put 'A.W' in temp message

ADDA_Source_EA

No

D2 == 7

Yes

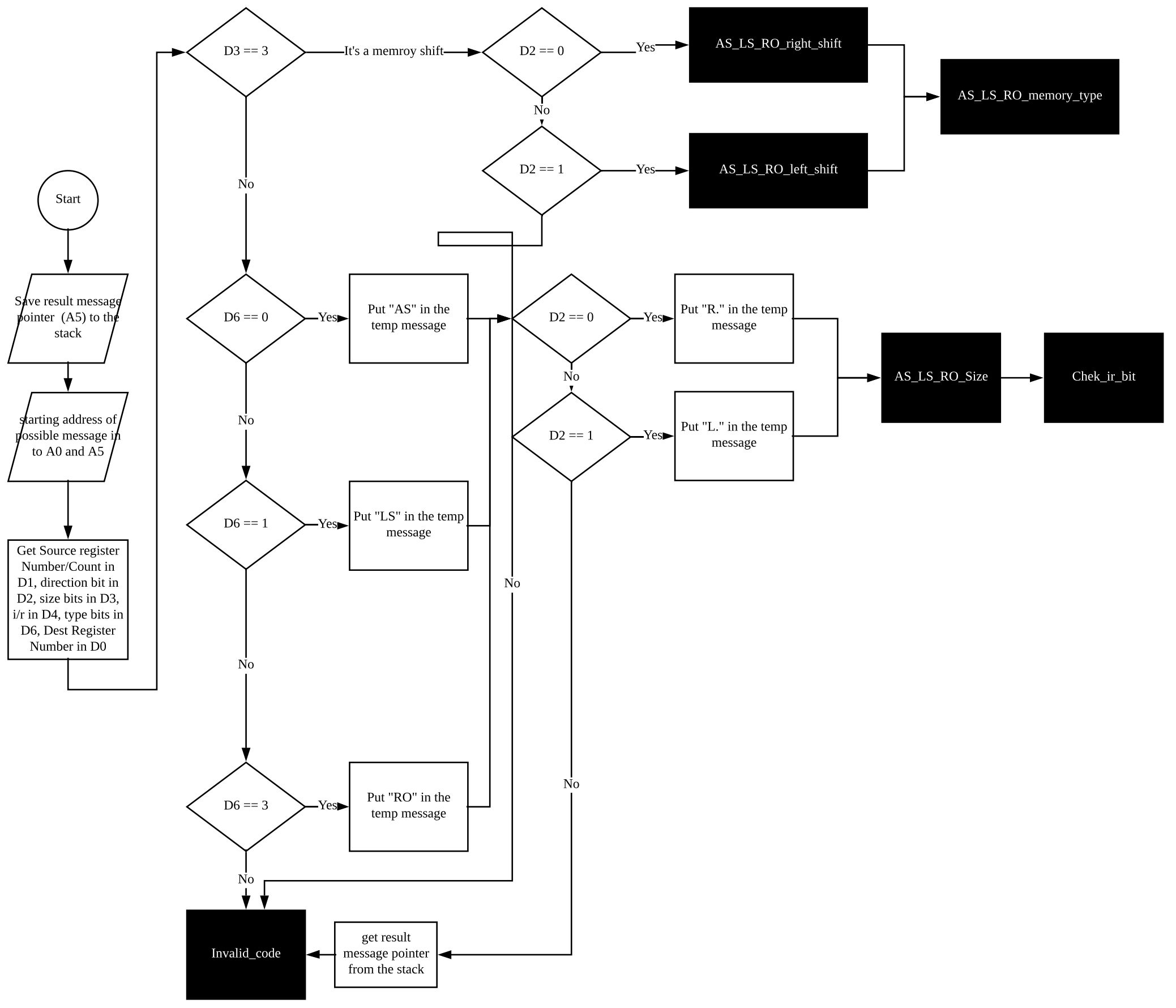
Put 'A.L' in temp message

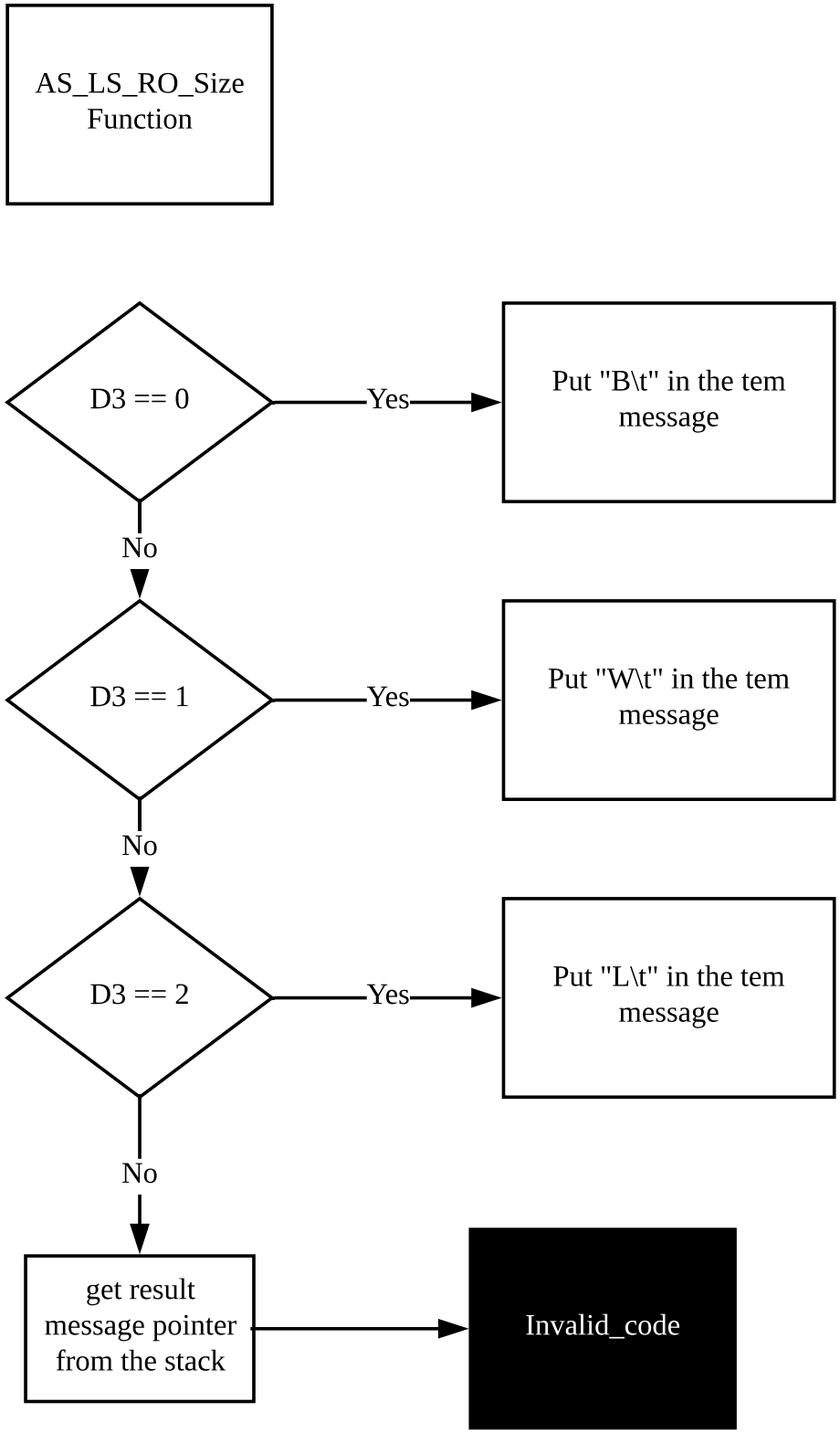
No

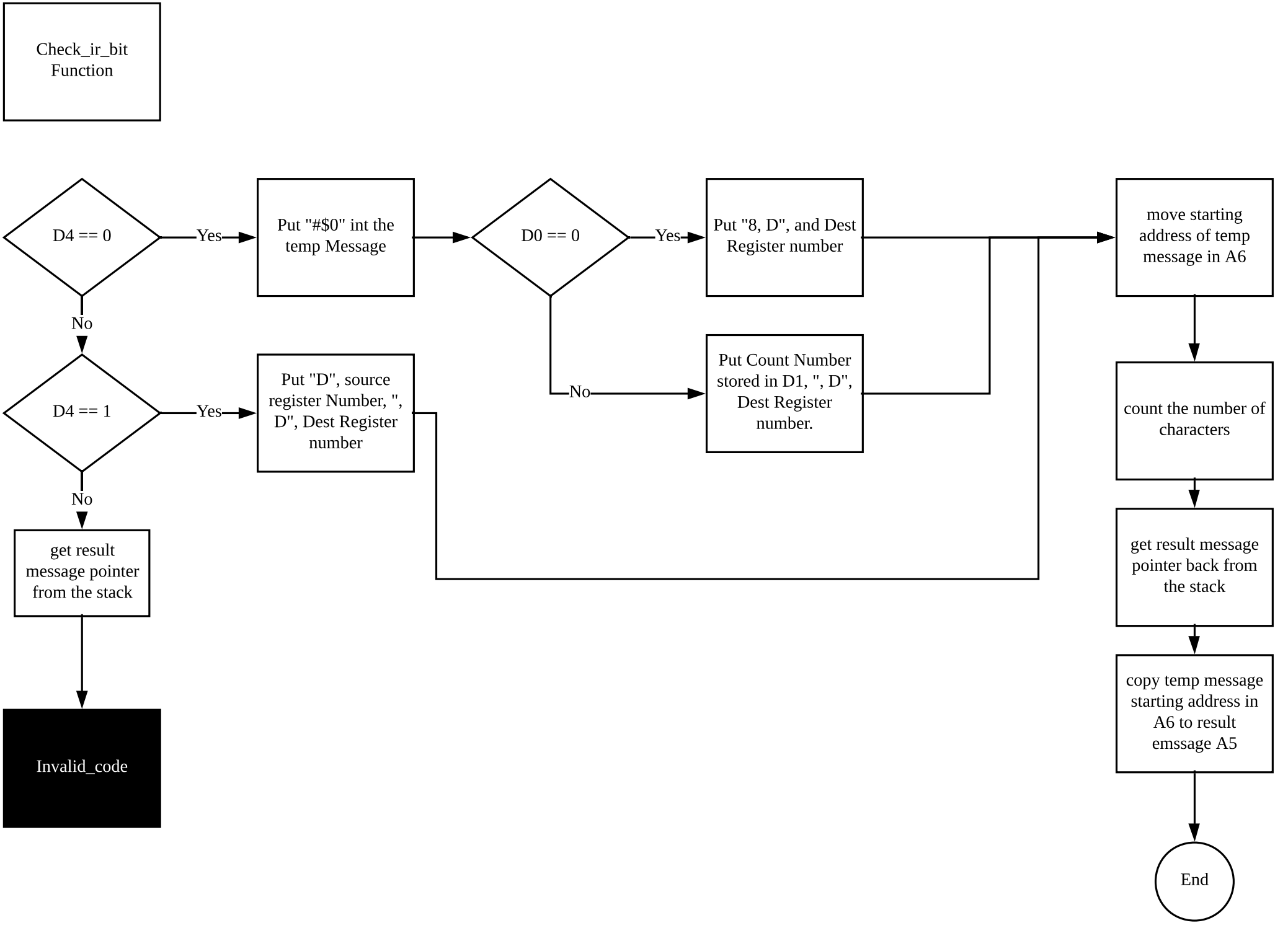
get result message pointer from the stack

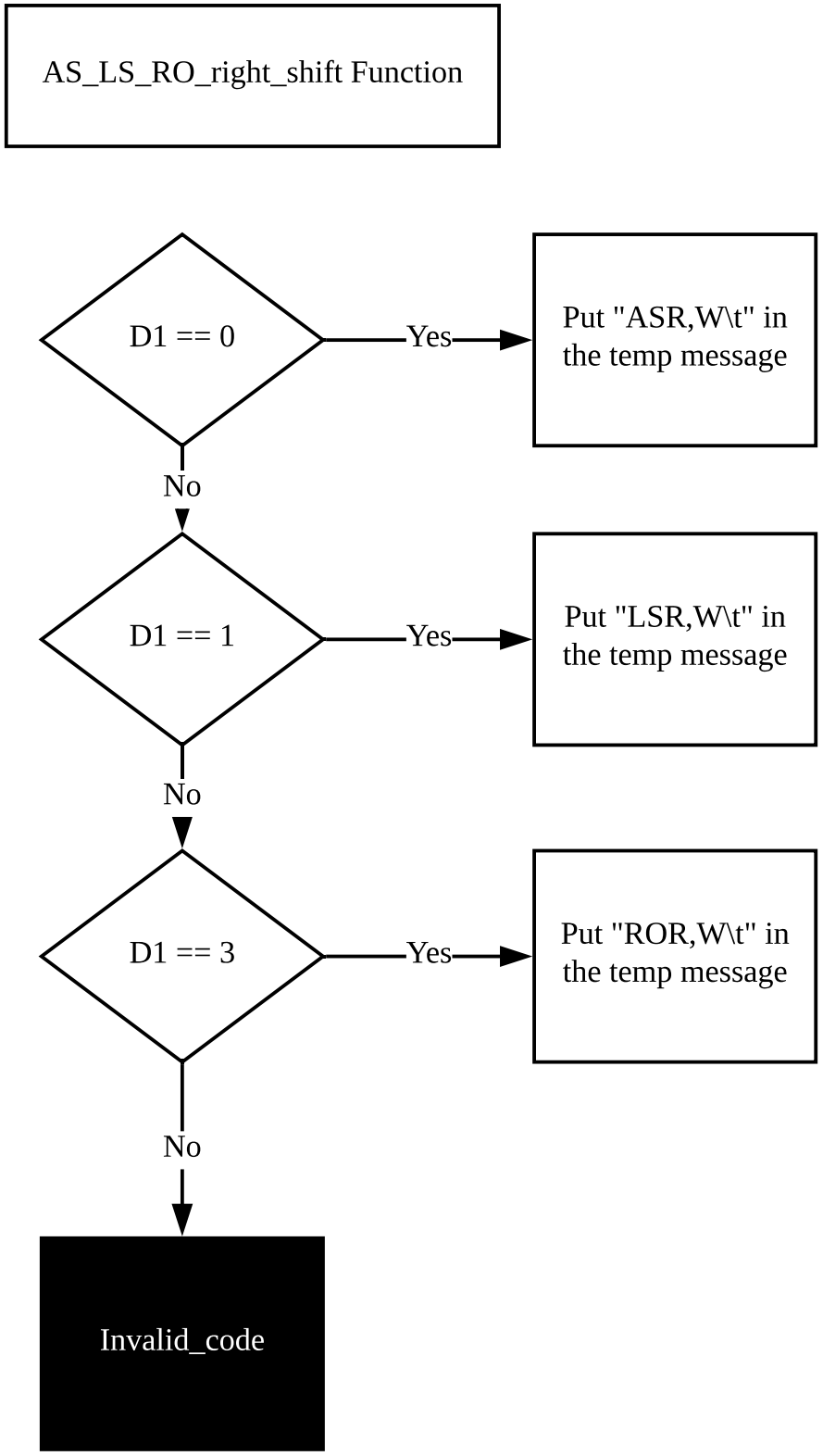
Invalid_code

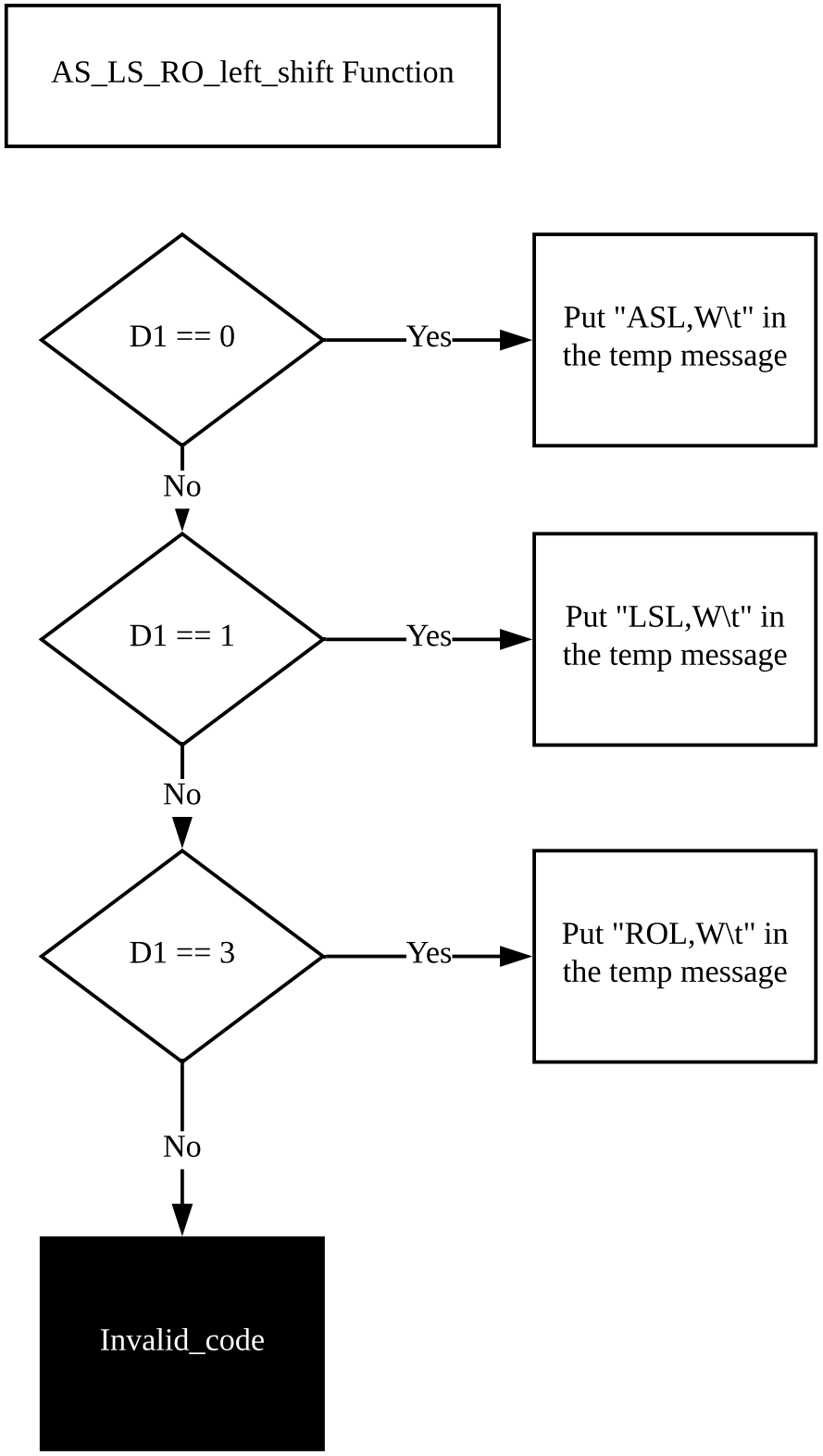
AS_LS_RO_LR_BWL Function

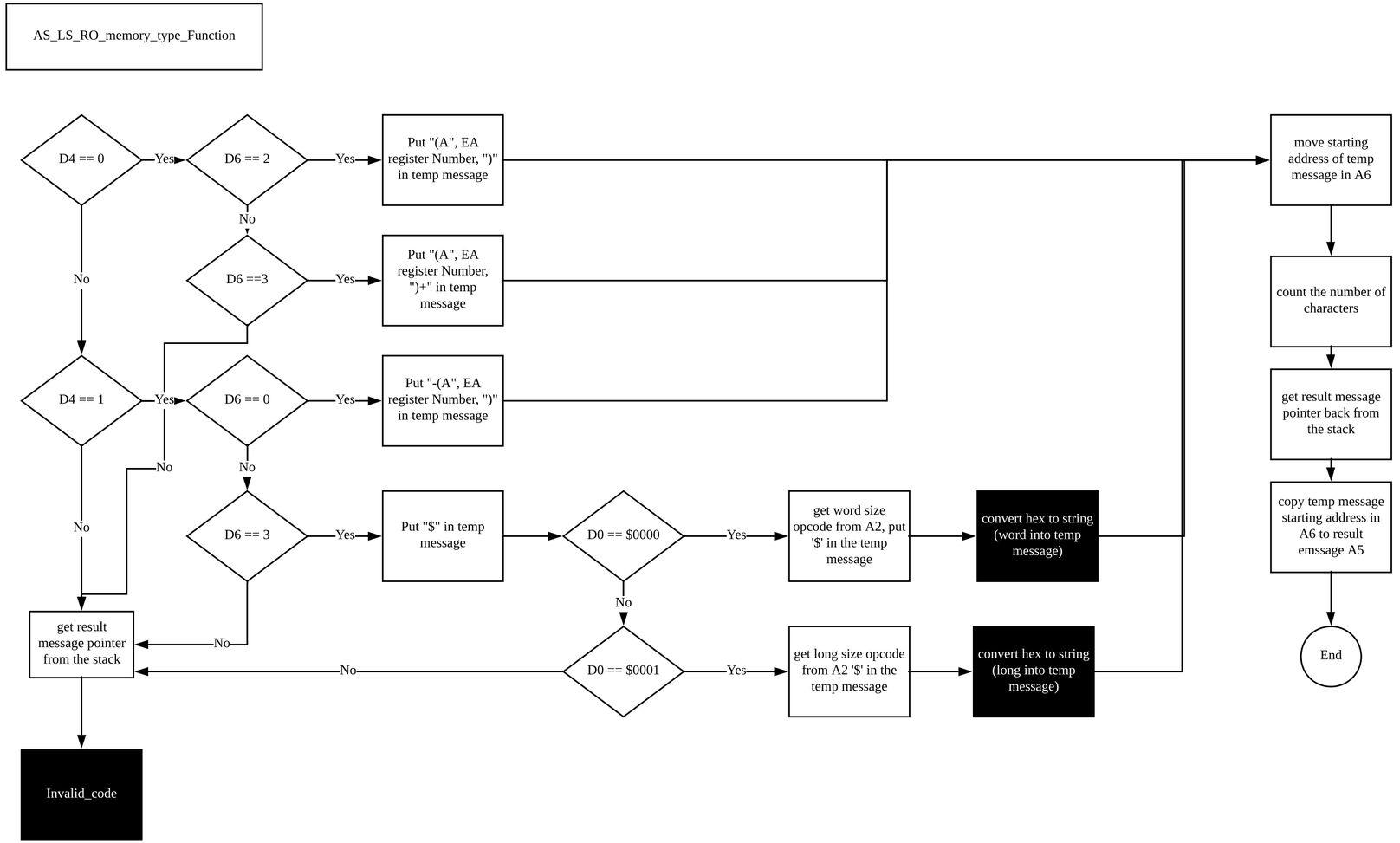


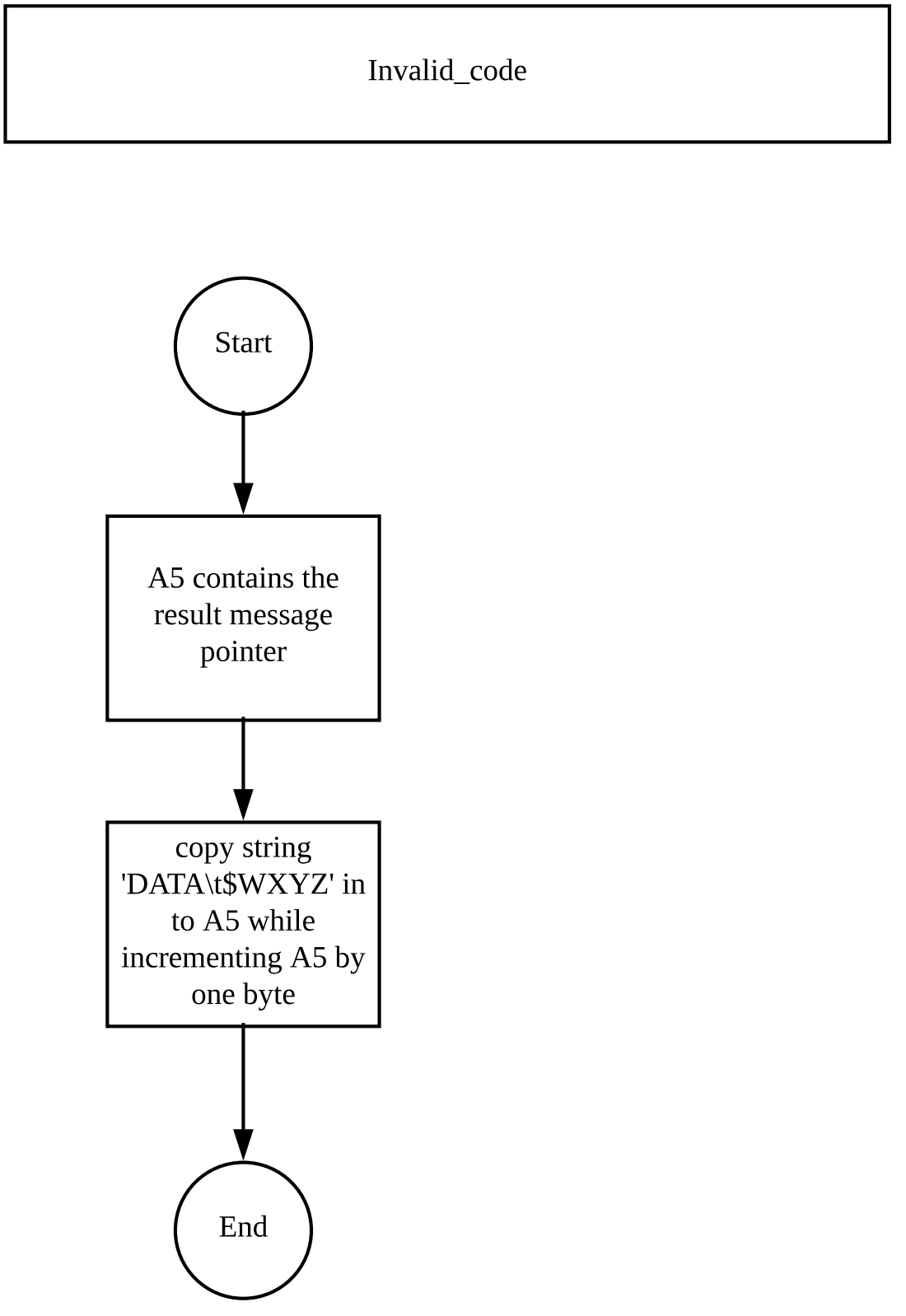

AS_LS_RO_Size Function
D3 == 0
Yes
Put "B\t" in the tem message
No
D3 == 1
Yes
Put "W\t" in the tem message
No
D3 == 2
Yes
Put "L\t" in the tem message
No
get result message pointer from the stack
Invalid_code

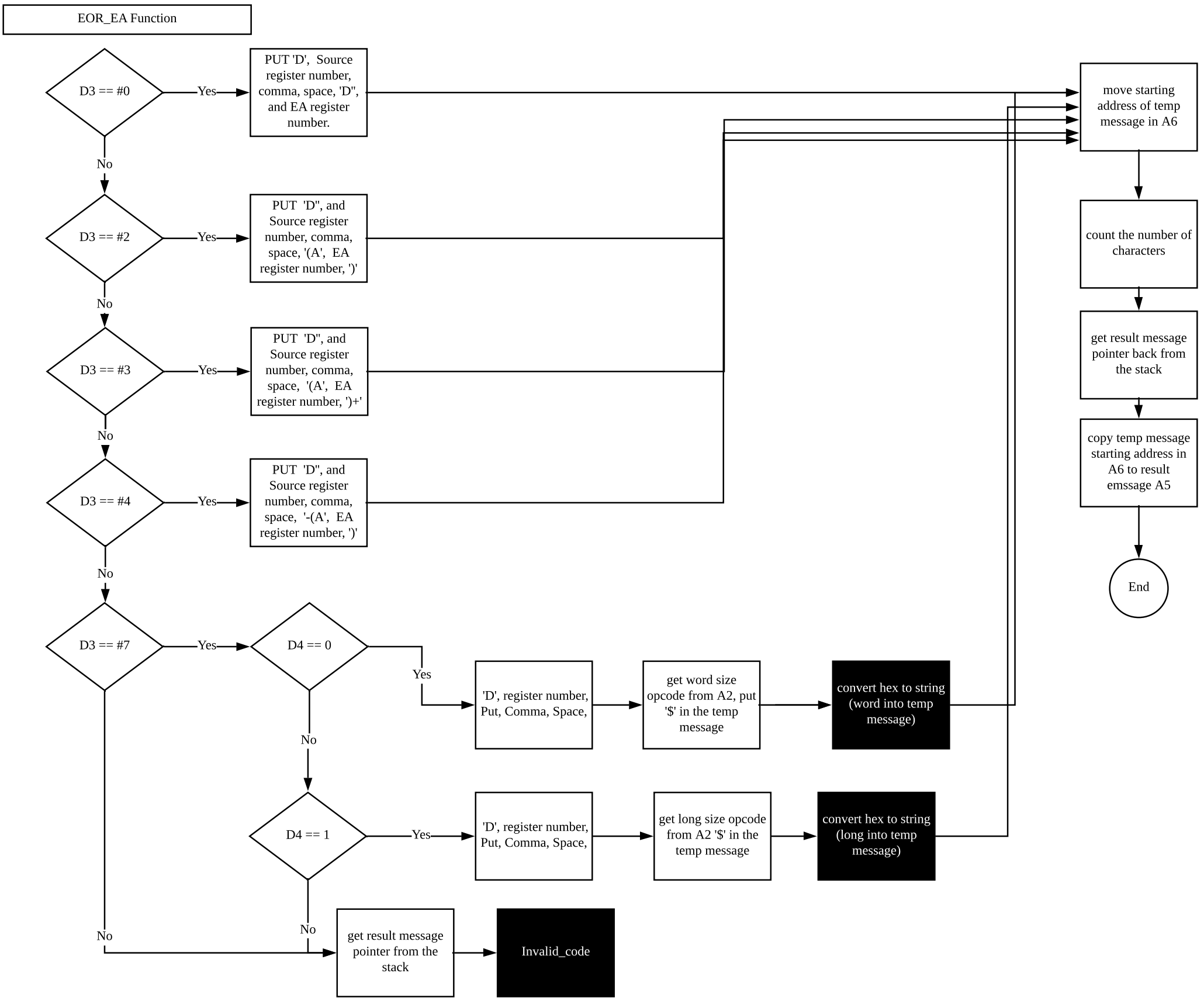

Check_ir_bit Function
D4 == 0
Yes
Put "#$0" int the temp Message
D0 == 0
Yes
Put "8, D", and Dest Register number
No
Put Count Number stored in D1, ", D", Dest Register number.
No
D4 == 1
Yes
Put "D", source register Number, ", D", Dest Register number
No
get result message pointer from the stack
Invalid_code
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


AS_LS_RO_right_shift Function
D1 == 0
Yes
Put "ASR,W\t" in the temp message
No
D1 == 1
Yes
Put "LSR,W\t" in the temp message
No
D1 == 3
Yes
Put "ROR,W\t" in the temp message
No
Invalid_code


AS_LS_RO_left_shift Function
D1 == 0
Yes
Put "ASL,W\t" in the temp message
No
D1 == 1
Yes
Put "LSL,W\t" in the temp message
No
D1 == 3
Yes
Put "ROL,W\t" in the temp message
No
Invalid_code


AS_LS_RO_memory_type_Function
D4 == 0
Yes
D6 == 2
Yes
Put "(A", EA register Number, ")" in temp message
No
No
D6 ==3
Yes
Put "(A", EA register Number, ")+" in temp message
D4 == 1
Yes
D6 == 0
Yes
Put "-(A", EA register Number, ")" in temp message
No
No
No
D6 == 3
Yes
Put "$" in temp message
D0 == $0000
Yes
get word size opcode from A2, put '$' in the temp message
convert hex to string (word into temp message)
No
get result message pointer from the stack
No
D0 == $0001
Yes
get long size opcode from A2 '$' in the temp message
convert hex to string (long into temp message)
No
Invalid_code
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


Invalid_code
Start
A5 contains the result message pointer
copy string 'DATA\t$WXYZ' in to A5 while incrementing A5 by one byte
End


EOR_EA Function
D3 == #0
Yes
PUT 'D', Source register number, comma, space, 'D", and EA register number.
No
D3 == #2
Yes
PUT 'D", and Source register number, comma, space, '(A', EA register number, ')'
No
D3 == #3
Yes
PUT 'D", and Source register number, comma, space, '(A', EA register number, ')+'
No
D3 == #4
Yes
PUT 'D", and Source register number, comma, space, '-(A', EA register number, ')'
No
D3 == #7
Yes
D4 == 0
Yes
'D', register number, Put, Comma, Space,
get word size opcode from A2, put '$' in the temp message
convert hex to string (word into temp message)
No
D4 == 1
Yes
'D', register number, Put, Comma, Space,
get long size opcode from A2 '$' in the temp message
convert hex to string (long into temp message)
No
No
get result message pointer from the stack
Invalid_code
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End

CMP_EA Function

D3 == #0 — Yes → PUT 'D', EA register number, comma, space, 'D', and Source register number.

D3 == #1 — Yes → PUT 'A', EA Register Number, comma, space, 'D', and Source register number.

D3 == #2 — Yes → PUT '(A', EA register number, ')' comma, space, 'D', and Source register number.

D3 == #3 — Yes → PUT '(A', EA register number, ')+', comma, space, 'D', and Source register number.

D3 == #4 — Yes → PUT '-(A', EA register number, ')', comma, space, 'D', and Source register number.

D3 == #7 — Yes → D4 == 0 — Yes → get word size opcode from A2, put '$' in the temp message → convert hex to string (word into temp message) → Put, Comma, Space, 'D', register number

D4 == 0 — No → D4 == 1 — Yes → get long size opcode from A2 '$' in the temp message → convert hex to string (long into temp message) → Put, Comma, Space, 'D', register number

D4 == 1 — No → D4 == 4 — Yes → D2 == 0 — No → D2 == 1 — No → D2 == 2 — Yes → put '$' in the temp message & get opcode from A2 (either word or long size) → convert hex to string either long or hex

D2 == 2 — No → get result message pointer from the stack

D4 == 4 — No → get result message pointer from the stack

D2 == 0 — No → Invalid_code

D2 == 1 — No → put '#$' in the temp message while incrementing A5 by one byte → get next word size opcode from A2 → Convert hex immediate hex value to string and put in the temp message

D2 == 0 — No → put '#$' in the temp message while incrementing A5 by one byte → get next word size opcode from A2 → Check if first 4bits == $0 — No → get result message pointer from the stack

Check if first 4bits == $0 — Yes → Convert hex immediate hex value to string and put in the temp message

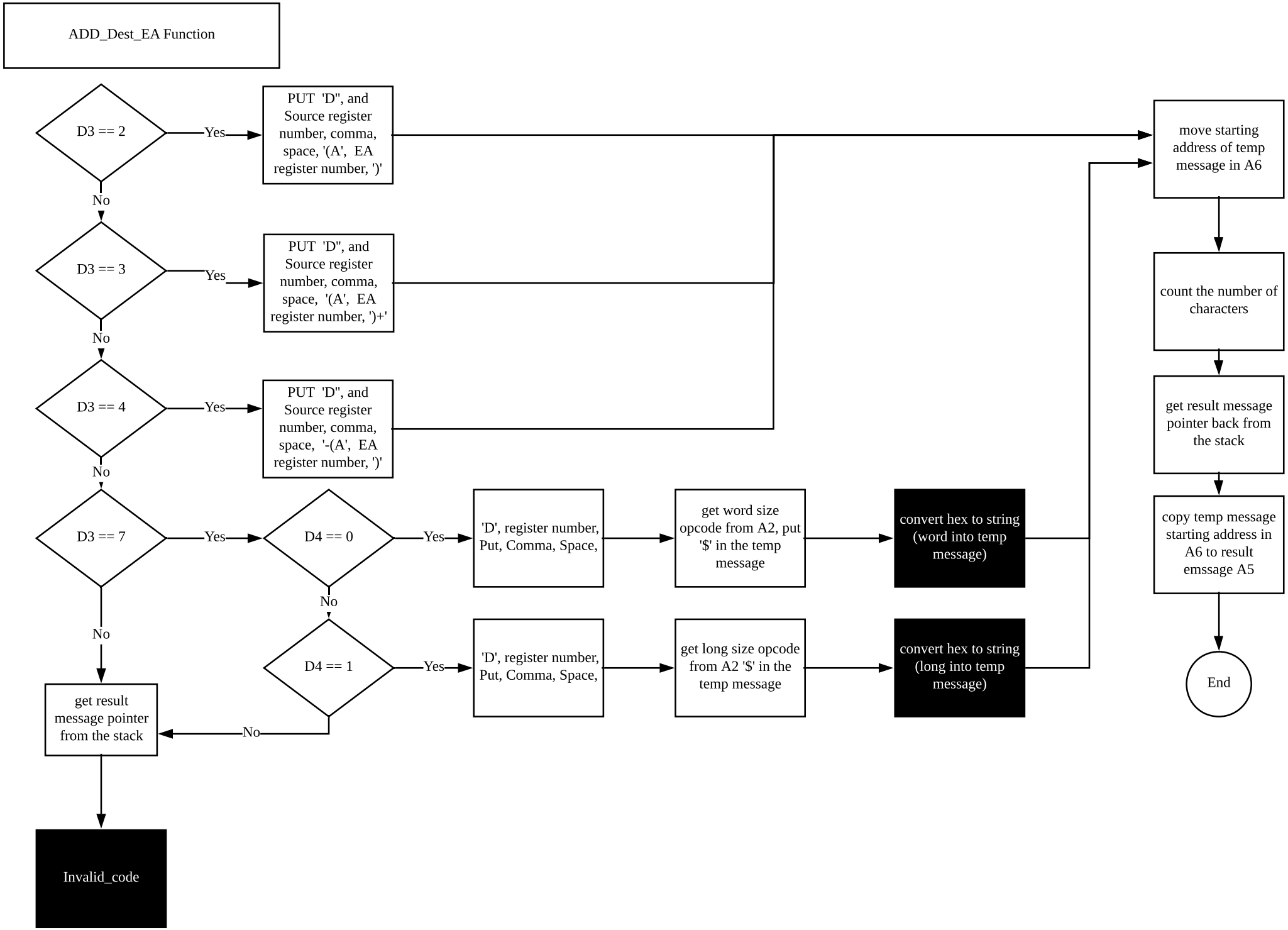
Put, Comma, Space, 'D', register number → move starting address of temp message in A6

move starting address of temp message in A6 → count the number of characters → get result message pointer back from the stack → copy temp message starting address in A6 to result emssage A5 → End

convert hex to string either long or hex → Put Comma, Space, 'D', register number → move starting address of temp message in A6

Add_Source_EA Function

D3 == 0
Yes
Put 'D', EA register number, comma, space, 'D', Source Register Number.
No
D3 == 1
Yes
Put 'A', EA register number, ', D', Source Register Number.
No
D3 == 2
Yes
Put '(A', EA register number, '), D', Source Register Number.
No
D3 == 3
Yes
Put '(A', EA register number, ')+, D', Source Register Number.
No
D3 == 4
Yes
Put '-(A', EA register number, '), D', Source Register Number.
No
D3 == 7
Yes
D4 == 0
Yes
get word size opcode from A2, put '$' in the temp message
convert hex to string (word into temp message)
No
D4 == 1
Yes
get long size opcode from A2 '$' in the temp message
convert hex to string (long into temp message)
No
D4 == 4
Yes
put '#$' in the temp message while incrementing A5 by one byte
get next word size opcode from A2
Convert hex immediate hex value to string and put in the temp message
No
D2 == 0
Yes
put '#$' in the temp message while incrementing A5 by one byte
get next word size opcode from A2
Check if first 4bits == $0
Yes
Convert hex immediate hex value to string and put in the temp message
No
D2 == 1
Yes
put '#$' in the temp message while incrementing A5 by one byte
get next word size opcode from A2
No
D2 == 2
Yes
put 'L\t#$' in the temp message while incrementing A5 by one byte
get next long size opcode from Address A2
No
Invalid_code
No
get result message pointer from the stack
No
'D', register number, Put, Comma, Space,
Put Comma, Space, 'D', register number
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End


ADD_Dest_EA Function
D3 == 2
Yes
PUT 'D", and Source register number, comma, space, '(A', EA register number, ')'
No
D3 == 3
Yes
PUT 'D", and Source register number, comma, space, '(A', EA register number, ')+'
No
D3 == 4
Yes
PUT 'D", and Source register number, comma, space, '-(A', EA register number, ')'
No
D3 == 7
Yes
D4 == 0
Yes
'D', register number, Put, Comma, Space,
get word size opcode from A2, put '$' in the temp message
convert hex to string (word into temp message)
No
D4 == 1
Yes
'D', register number, Put, Comma, Space,
get long size opcode from A2 '$' in the temp message
convert hex to string (long into temp message)
No
get result message pointer from the stack
No
Invalid_code
move starting address of temp message in A6
count the number of characters
get result message pointer back from the stack
copy temp message starting address in A6 to result emssage A5
End

AddA_Source_EA Function

D3 == 0 — Yes → Put 'D', EA register number, ', A', Source Register Number.
No
D3 == 1 — Yes → Put 'A', EA register number, ', A', Source Register Number.
No
D3 == 2 — Yes → Put '(A', EA register number, '), A', Source Register Number.
No
D3 == 3 — Yes → Put '(A', EA register number, ')+, A', Source Register Number.
No
D3 == 4 — Yes → Put '-(A', EA register number, '), A', Source Register Number.
No
D3 == 7 — Yes → D4 == 0 — Yes → get word size opcode from A2, put '$' in the temp message → convert hex to string (word into temp message) → Put, Comma, Space, 'A', register number

D4 == 0 — No → D4 == 1 — Yes → get long size opcode from A2 '$' in the temp message → convert hex to string (long into temp message) → Put, Comma, Space, 'A', register number

D4 == 1 — No → D4 == 4 — No → D2 == 0 — Yes → put '#$' in the temp message while incrementing A5 by one byte → get next word size opcode from A2 → Check if first 4bits == $0 — Yes → Convert hex immediate hex value to string and put in the temp message → Put Comma, Space, 'A', register number

D3 == 7 — No → get result message pointer from the stack → Invalid_code

D4 == 4 — No → get result message pointer from the stack

Check if first 4bits == $0 — No → get result message pointer from the stack

D2 == 0 — No → D2 == 1 — No → put '#$' in the temp message while incrementing A5 by one byte → get next word size opcode from A2

D2 == 1 — Yes → D2 == 2 — Yes → put '$' in the temp message & get opcode from A2 (either word or long size) → convert hex to string either long or hex → Put Comma, Space, 'A', register number

Put Comma, Space, 'A', register number → move starting address of temp message in A6 → count the number of characters → get result message pointer back from the stack → copy temp message starting address in A6 to result emssage A5 → End