



# CPU Toolchain Launch Postmortem

Greg Bedwell



x86-64 AMD “Jaguar”  
8-core CPU

1.84 TFLOPS AMD Radeon™  
based GPU

8GB GDDR5 RAM

## Developer Toolchain for



Paul T. Robinson  
Sony Computer Entertainment  
LLVM Dev Meeting, 7 Nov 2013



## Agenda

- **PlayStation®4 – Info for game teams**
- Why Clang?
- Special Considerations
- Hacking on Clang/LLVM
- Now and the Future

<http://llvm.org/devmtg/2013-11/>

# postmortem *noun*

*“an analysis or discussion of an event after it is over”*

Now that we have successfully launched PlayStation®4 it is a good time to look back on our initial period of development up to that point

<http://www.merriam-webster.com/dictionary/postmortem>

*condensed*  
***First, some history...***

SN Systems Ltd. was founded in 1988 to provide development tools for the games industry

**1990** *Psy-Q*  
*16-bit home systems*



Psy-Q included a version of GCC that was highly customized for the needs of game developers

**1994** *Psy-Q*  
*PlayStation*<sup>®</sup>



Continued to provide GCC but started researching a proprietary compiler technology – “SNC”

# 2000 ProDG

PlayStation®2





Provided SNC as part of the ProDG suite of tools although GCC was also available

## **2004** *ProDG*

*PSP®*

*(PlayStation®Portable)*

Sony Computer  
Entertainment Inc.  
acquired SN Systems  
in 2005



Provided SNC as part of  
the ProDG suite of tools  
although GCC was also  
available

**2006** *ProDG*  
*PlayStation®3*



CPU Compiler is SNC

**2011**

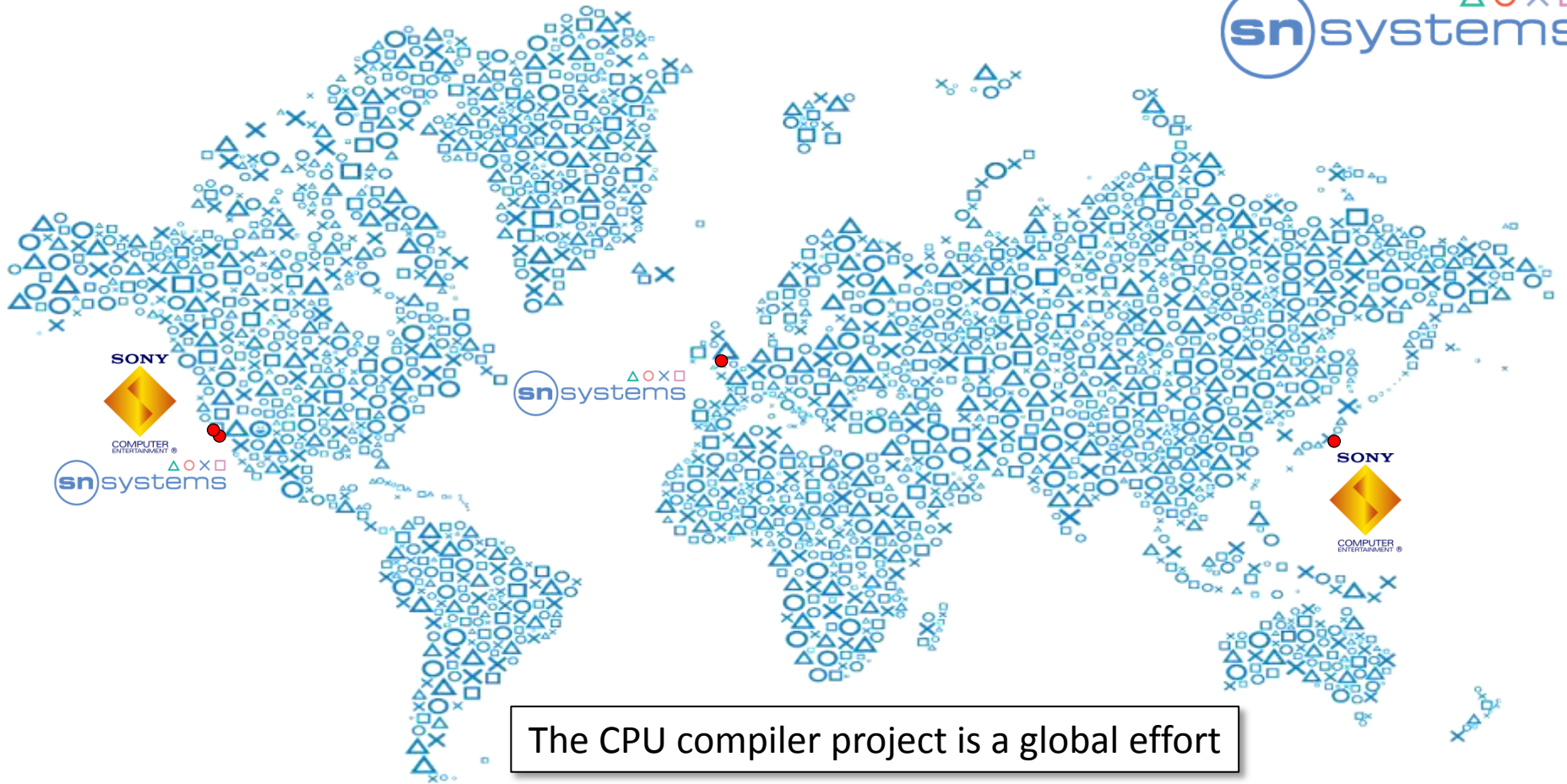
*PlayStation®Vita*



CPU Compiler is Clang

**2013**

*PlayStation®4*



The CPU compiler project is a global effort

# ***Builds and build systems*** (and test systems)

```
Administrator: C:\windows\system32\cmd.exe - python regression_suite.py --skip-run --platform PS4 --cflags="-O2 -g" --num-threads=100
```

```
SCE CPU Compiler Regression Suite (press Q to quit)
```













```
[7s] vector4_class [7s] Param_Passing_types5 [7s] Param_Passing_types22 [7s] Param_Passing_types44
[7s] random_struct_layouts_1 [7s] Param_Passing_types44 [7s] Param_Passing_types21 [7s] Param_Passing_types44
[7s] random_struct_layouts_2 [7s] Param_Passing_types43 [7s] Param_Passing_types28 [7s] Param_Passing_types44
[7s] random_struct_layouts_3 [7s] Param_Passing_types42 [7s] Param_Passing_types27 [7s] Param_Passing_types44
[7s] intrinsics-tester [7s] Param_Passing_types41 [7s] Param_Passing_types19 [7s] Param_Passing_types44
[7s] SCEE_Devstation13_fun_wi [7s] Param_Passing_types40 [7s] Param_Passing_types18 [7s] Param_Passing_types44
[7s] SCEE_Devstation13_everysd [7s] Param_Passing_types4 [7s] Param_Passing_types17 [7s] Param_Passing_types44
[7s] SCEE_Devstation13_fun_wi [7s] Param_Passing_types39 [7s] Param_Passing_types16 [7s] Param_Passing_types44
[7s] SCEA_Devcon2013_ext_vect [7s] Param_Passing_types38 [7s] Param_Passing_types14 [7s] Param_Passing_types44
[7s] RTTI [7s] Param_Passing_types37 [7s] Param_Passing_types15 [7s] Param_Passing_types44
[7s] Param_Passing_vector1 [7s] Param_Passing_types36 [7s] Param_Passing_types11 [7s] Param_Passing_types44
[7s] Param_Passing_variadic9 [7s] Param_Passing_types34 [7s] Param_Passing_types12 [7s] Param_Passing_types44
[7s] Param_Passing_variadic8 [7s] Param_Passing_types35 [7s] Param_Passing_types11 [7s] Param_Passing_types44
[7s] Param_Passing_variadic7 [7s] Param_Passing_types33 [7s] Param_Passing_types10 [7s] Param_Passing_types44
[7s] Param_Passing_variadic6 [7s] Param_Passing_types32 [7s] Param_Passing_types1 [7s] Param_Passing_types44
[7s] Param_Passing_variadic5 [7s] Param_Passing_types31 [7s] Param_Passing_structs1 [7s] Param_Passing_types44
[7s] Param_Passing_variadic4 [7s] Param_Passing_types30 [7s] Param_Passing_ansi1 [7s] Param_Passing_types44
[7s] Param_Passing_variadic3 [7s] Param_Passing_types3 [7s] Param_Passing_resources1 [7s] Param_Passing_types44
[7s] Param_Passing_variadic2 [7s] Param_Passing_types29 [7s] Param_Passing_reftypes1 [7s] Param_Passing_types44
[7s] Param_Passing_variadic1 [7s] Param_Passing_types28 [7s] Param_Passing_reftypes4 [7s] Param_Passing_types44
[7s] Param_Passing_types9 [7s] Param_Passing_types26 [7s] Param_Passing_reftypes2 [7s] Param_Passing_types44
[7s] Param_Passing_types8 [7s] Param_Passing_types27 [7s] Param_Passing_reftypes1 [7s] Param_Passing_types44
[7s] Param_Passing_types7 [7s] Param_Passing_types25 [7s] Param_Passing_reftypes1 [7s] Param_Passing_types44
[7s] Param_Passing_types6 [7s] Param_Passing_types24 [7s] Param_Passing_reftypes1 [7s] Param_Passing_types44
[7s] Param_Passing_types45 [7s] Param_Passing_types23 [7s] Param_Passing_noregisters1 [7s] Param_Passing_types44
```













```
==
Q:1116, R:100, P:22, F:0, IgP:0, IgF:0, ABORT:0
Creating test set for platform 'PS4'
Found 1238 tests (searched 2902 dirs)
```

We wanted to make use of all of our pre-existing test suites and test systems, which are shared across all targets

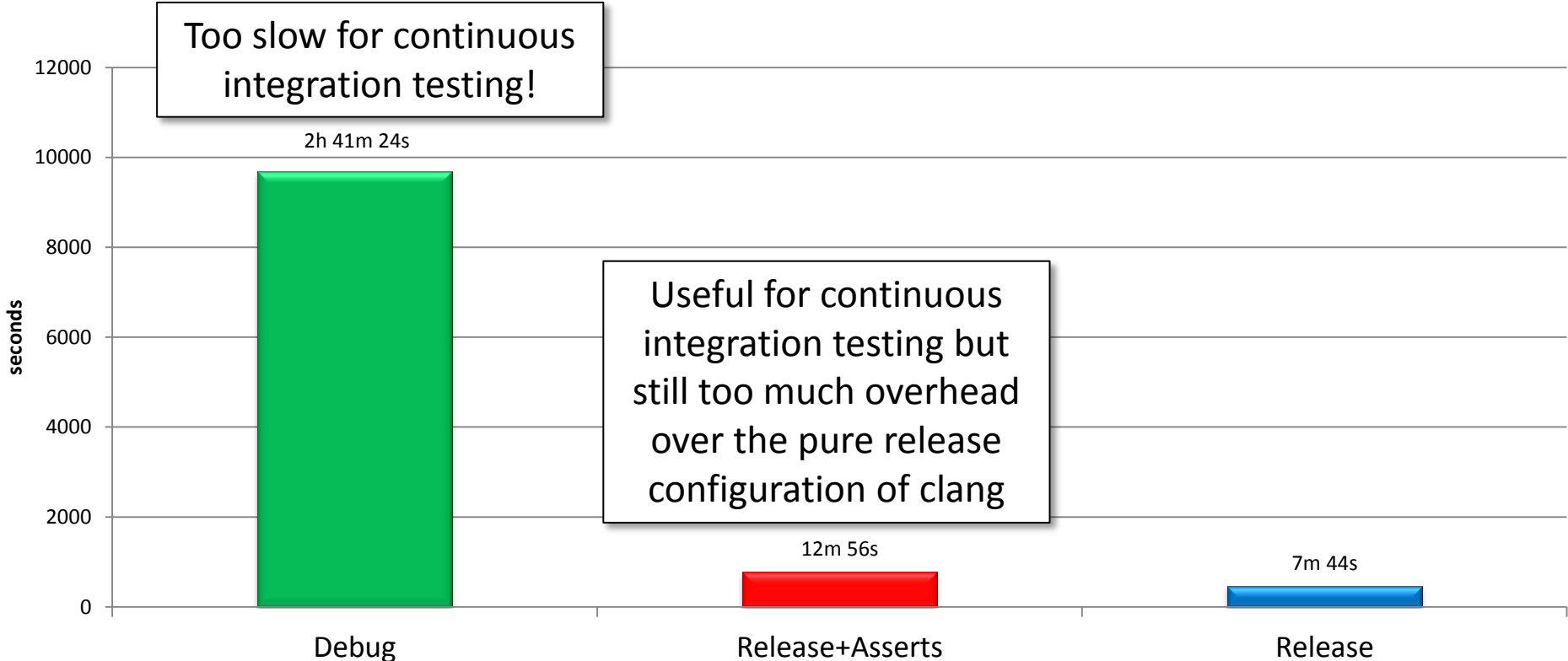
	Mar 07 04:11	Mar 05 12:13	Mar 04 18:53	Mar 04 16:25	Mar 01 03:40	Feb 28 17:13	Feb 27 17:13	Feb 26 13:22	Feb 24 21:39	Feb 24 18:51	Feb 14 22:39	Feb 14 17:42	Feb 14 15:13	Feb 11 23:37	Feb 11 15:18	
Newest ←																
	75%	100%	98%	98%	98%	98%	98%	98%	98%	98%	98%	98%	98%	97%	95%	
																
																












	Optimized	Assertions	Debug Info
Debug			
MinSizeRel			
Release			
RelWithDebInfo			

	Optimized	Assertions	Debug Info
Debug			
MinSizeRel			
Release			
RelWithDebInfo			

# Clang configuration effect on game build time












	Optimized	Assertions	Debug Info
Debug			
Checking			
Release			

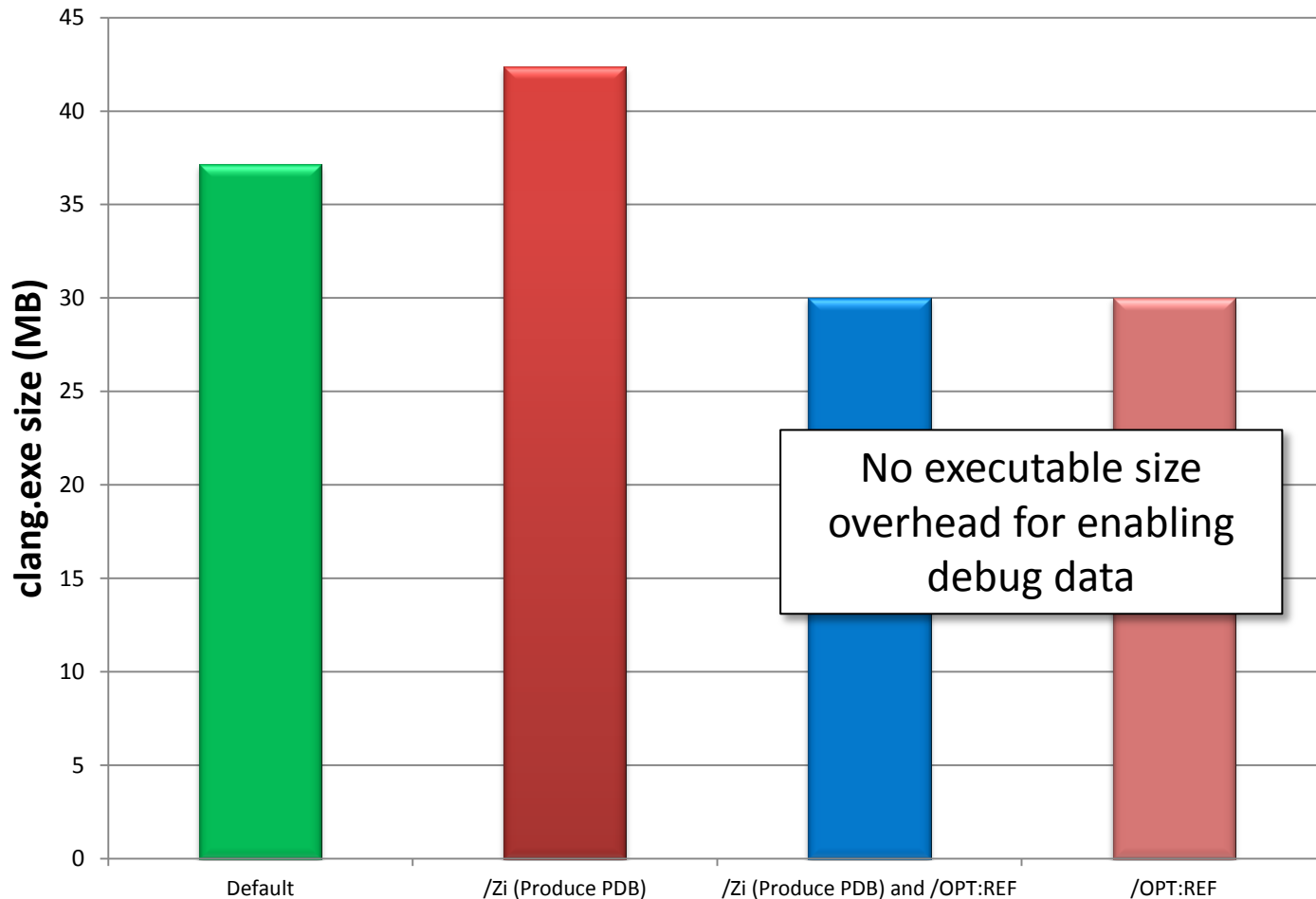
The same set of build configurations as we use for SNC










“Debug” for debugging

“Release” for releasing

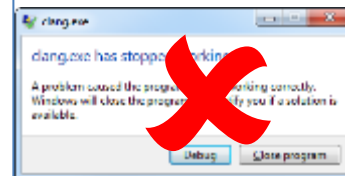
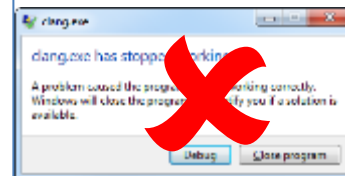
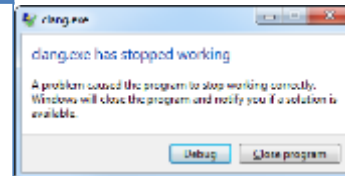
“Checking”  
(Release+Asserts) for continuous integration testing

	Optimized	Assertions	Debug Info
Debug			
Checking			
Release			



	Optimized	Assertions	Debug Info
Debug			
Checking			
Release			

	Optimized	Assertions	Debug Info
<b>Debug</b>	✗	✓	✓
<b>Checking</b>	✓	✓	✓
<b>Release</b>	✓	✗	✓



Suppress Windows crash dialog box for  
Checking and Release builds



# *Improving test coverage*

Things  
we  
know

Game code is  
usually

***BIG***

For a new platform, the  
amount of code that exists is

*small*

Things  
we  
know

Users value

***CORRECTNESS***

over all else

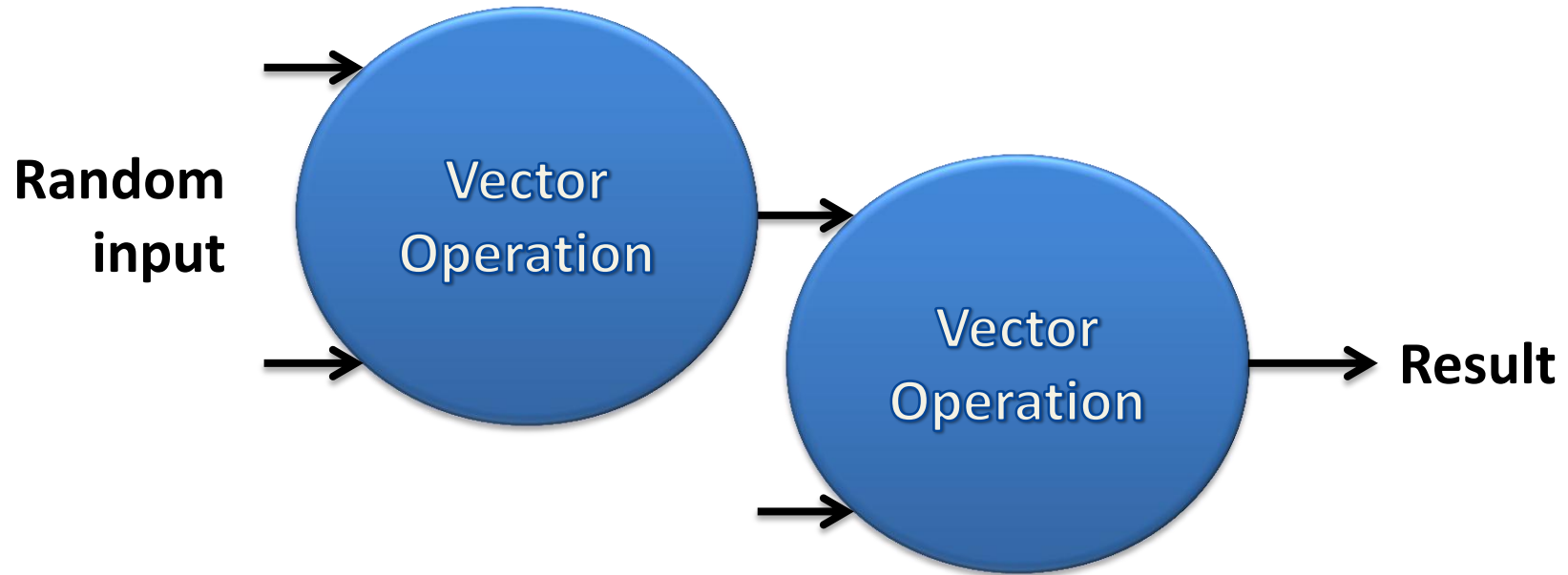
*Why write tests  
when I can write a test generator?*

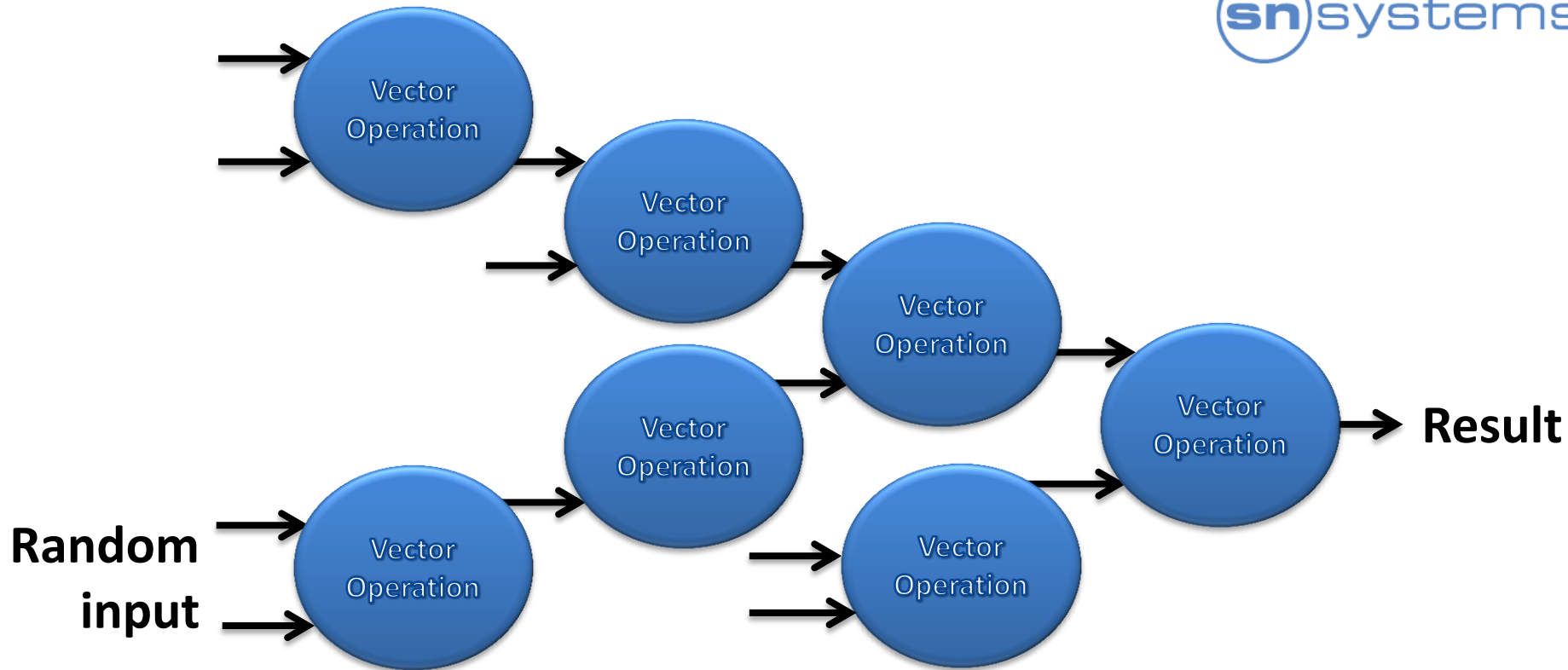
## *Why write tests*

*when I ~~can write a test generator?~~  
already have*

Generate random C++ tests using SIMD language extensions and intrinsics to increase test coverage

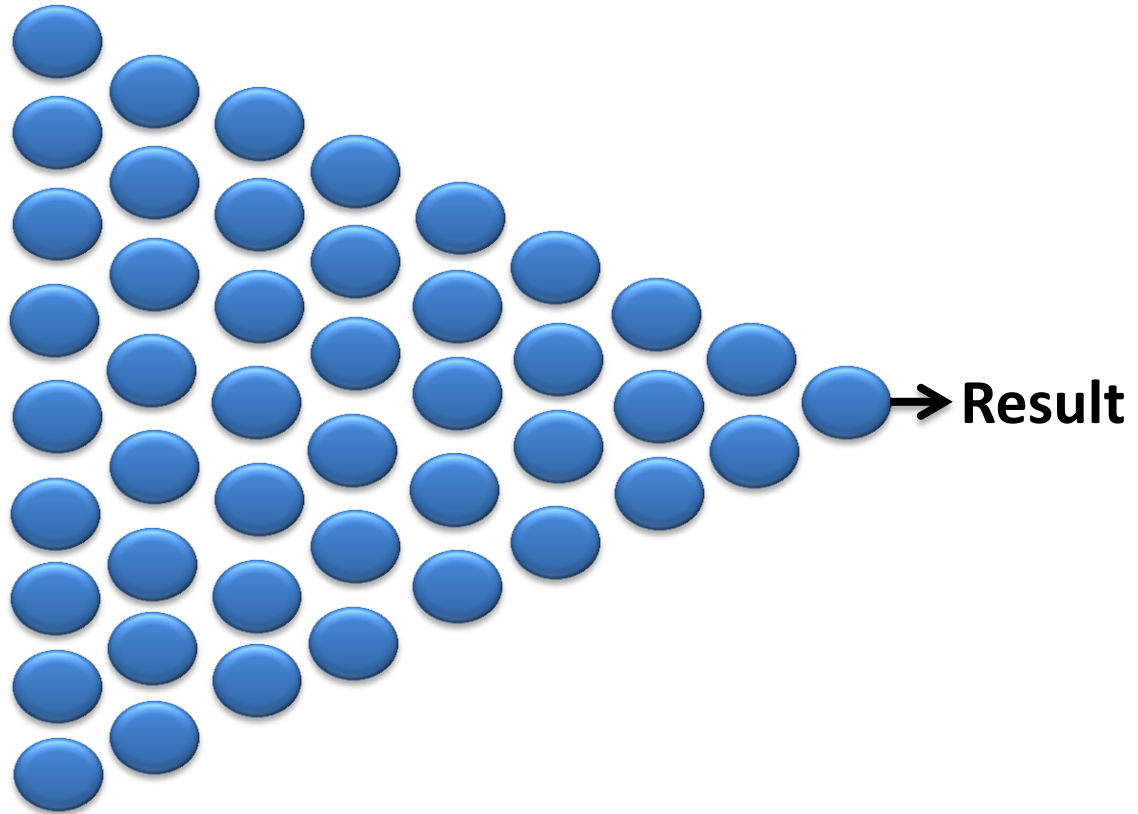


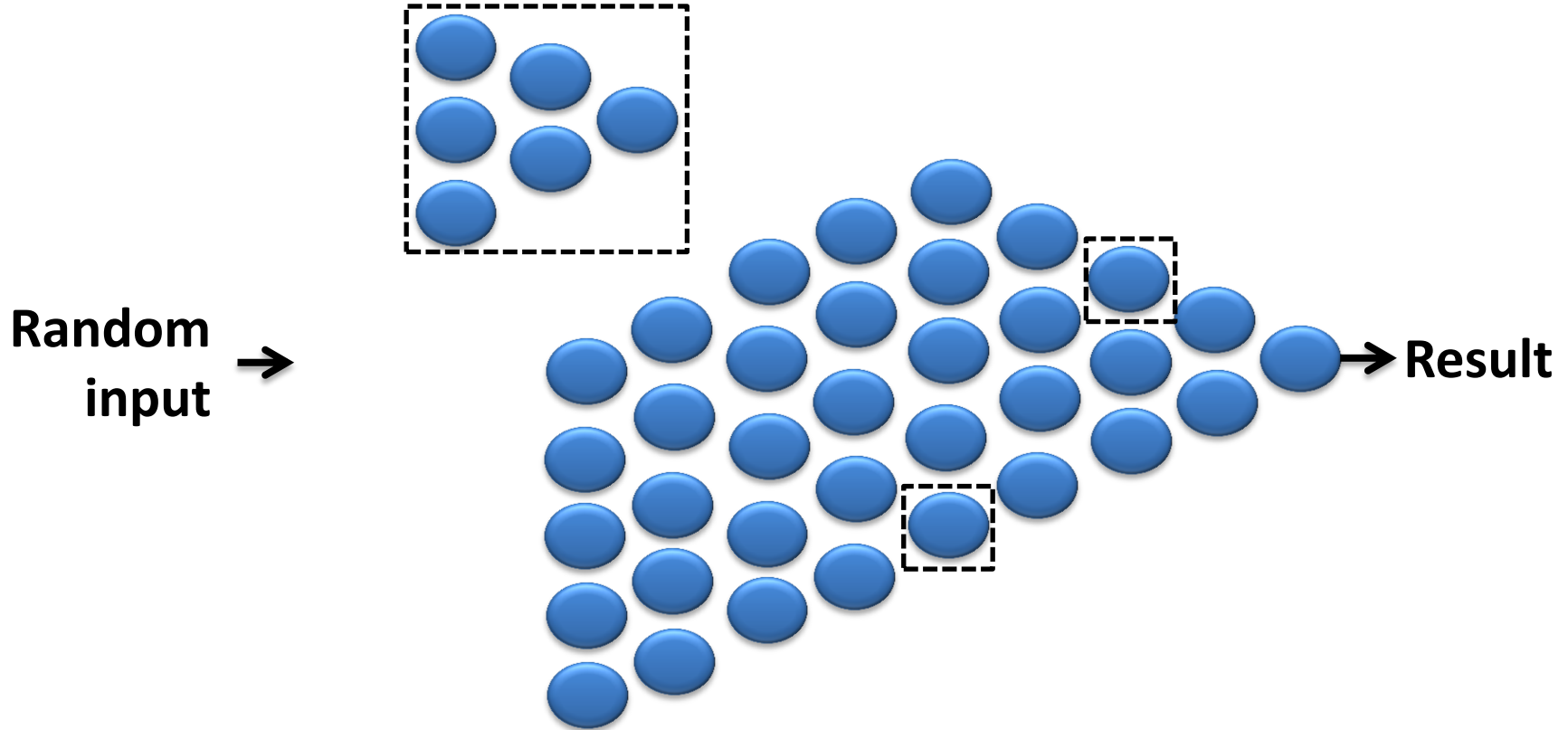


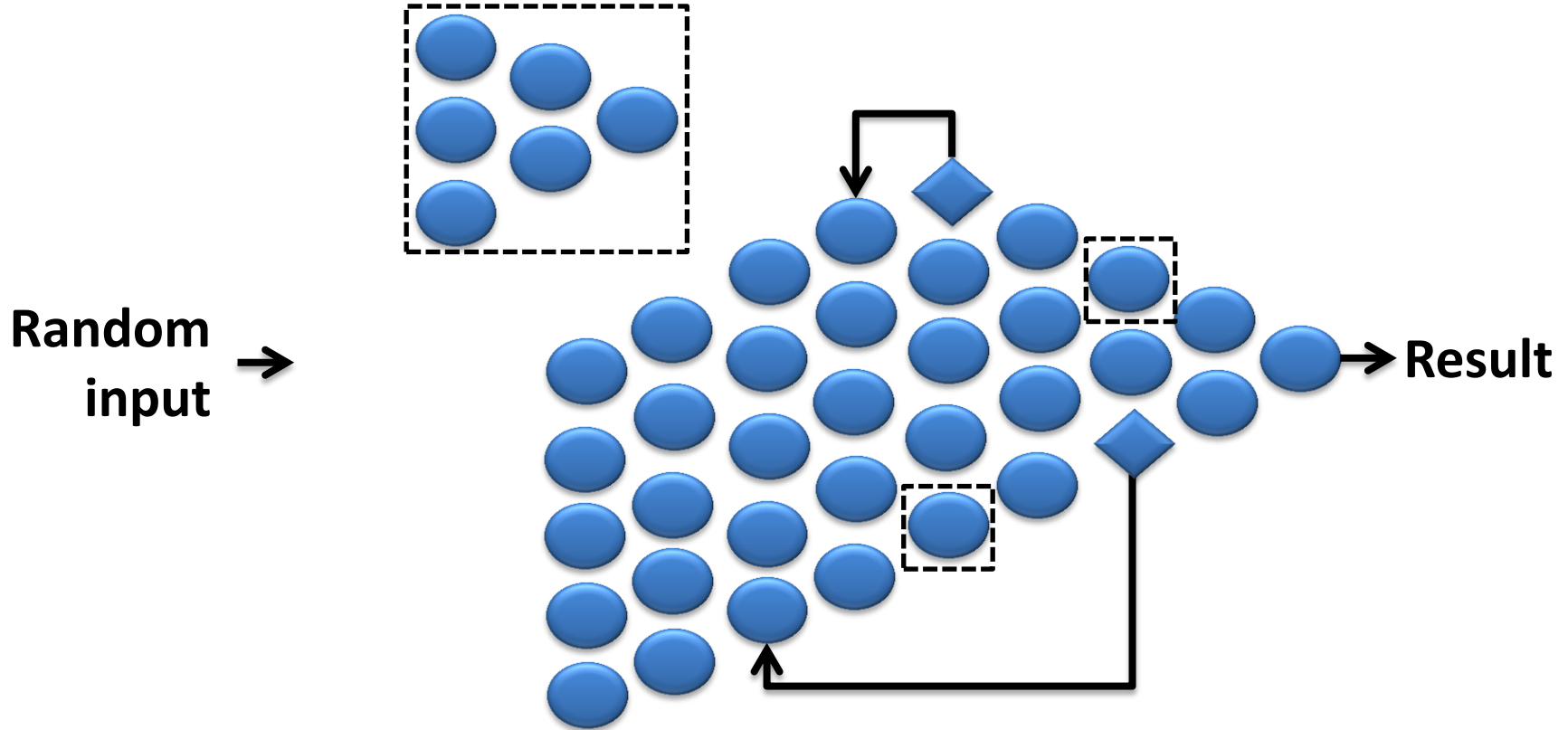




Random  
input →









Undefined behaviour  
makes runtime-  
behaviour random  
testing hard!



Solution: Use a 'safe' wrapper to make all undefined behaviour defined for the purpose of the test

# *Reducing optimization bugs*

# bugpoint

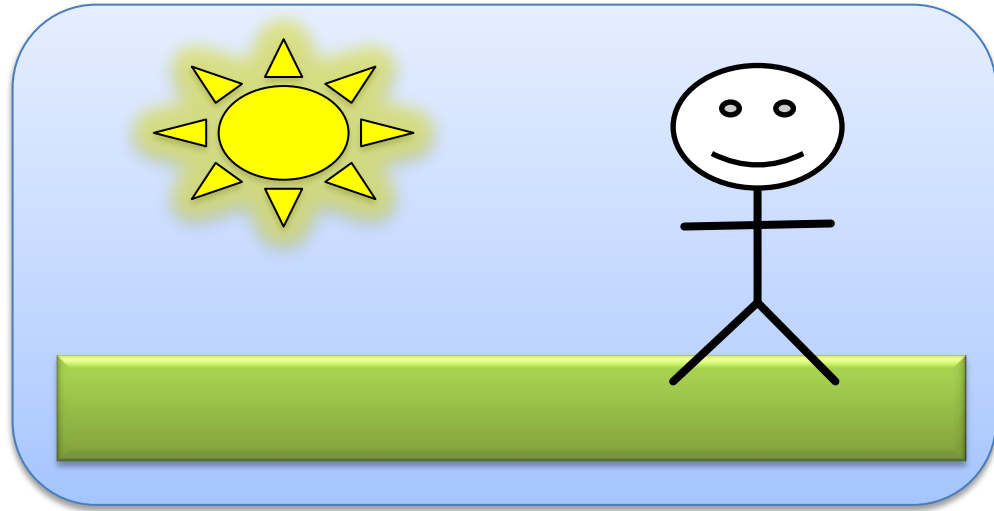
Windows

Clang integration

# *An alternative approach*



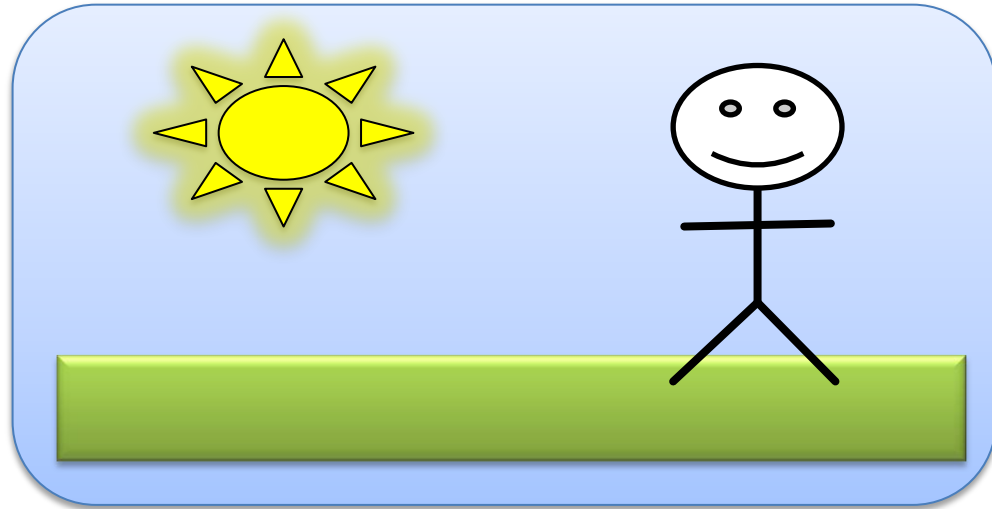
# SNC's max\_opts



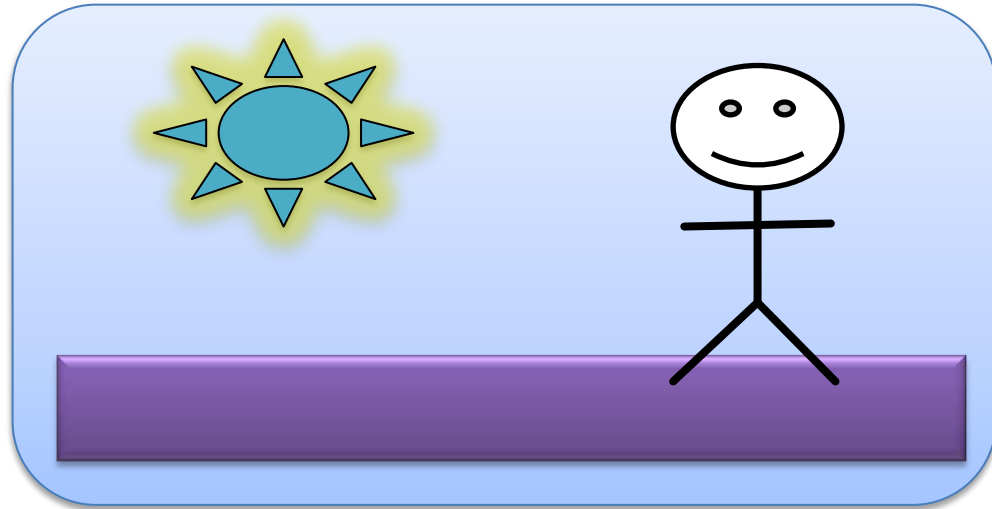
\*Not representative of actual visuals



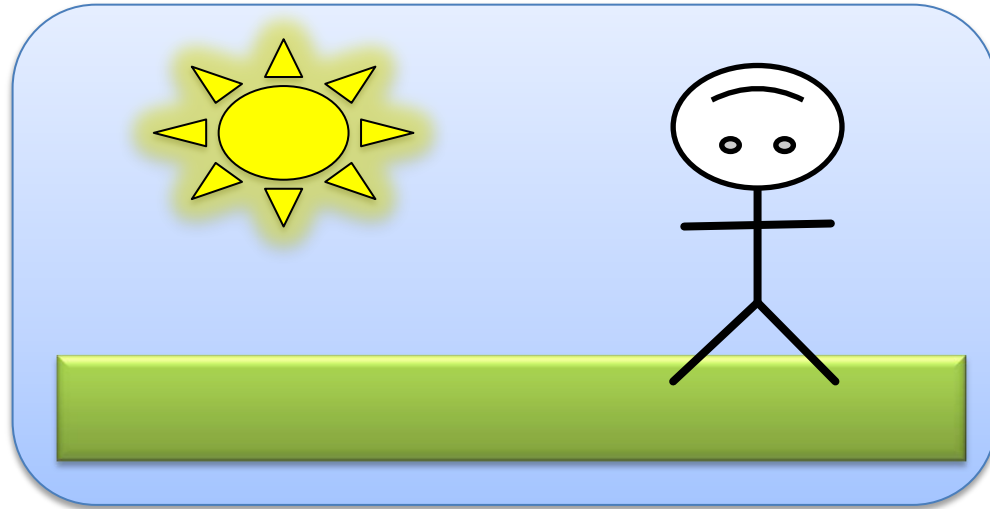
# SNC's max\_opts



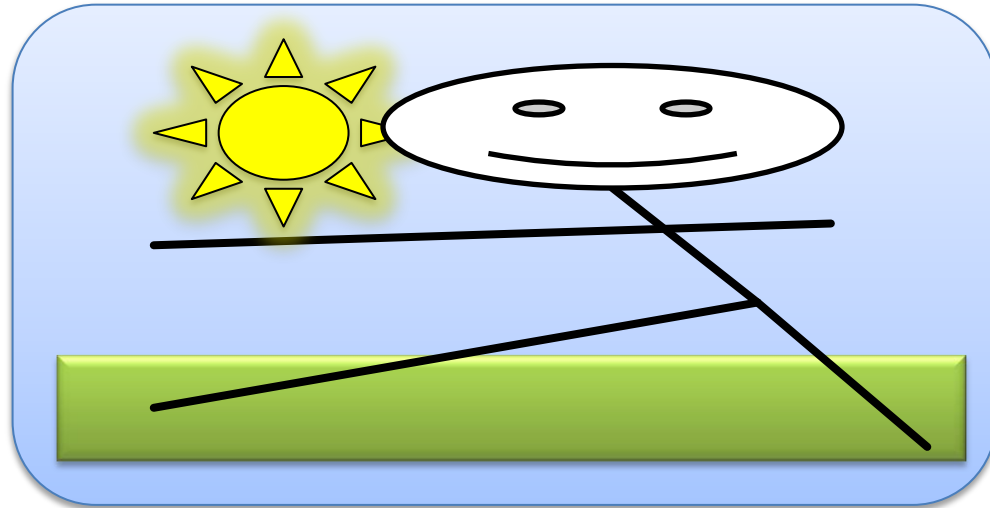
# SNC's max\_opts



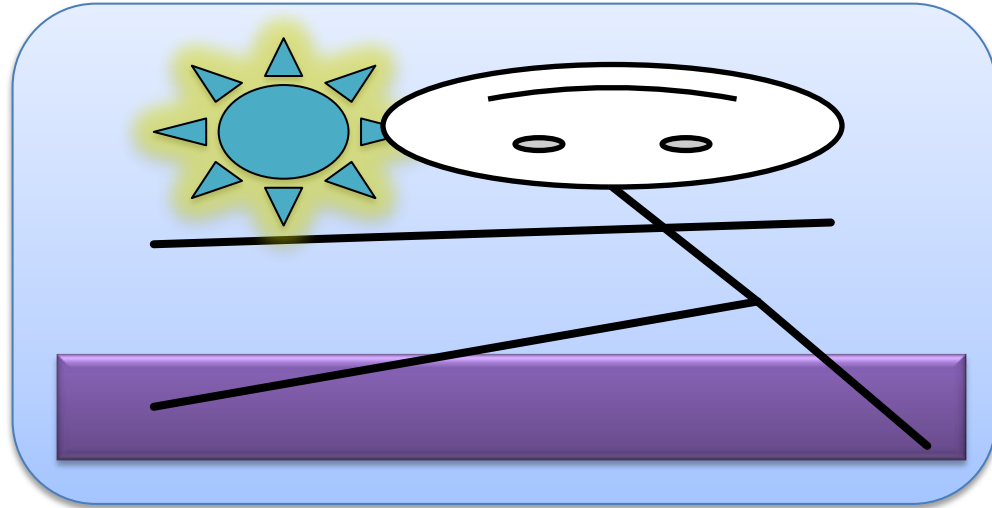
# SNC's max\_opts



# SNC's max\_opts



# SNC's max\_opts



SNC's  
max\_opts

## SNC optimizer

*SSA Form*

*Rule Based*

*Every transformation is guarded  
by a specific check*



SNC keeps an internal counter of the number of transformations it performs

```
if ( /* conditions match */ &&
    Opt_Enabled(permute_converted_to_opscalartovector) )
{
    Trace_Opt(permute_converted_to_opscalartovector);
    /* Perform optimization */
}
```

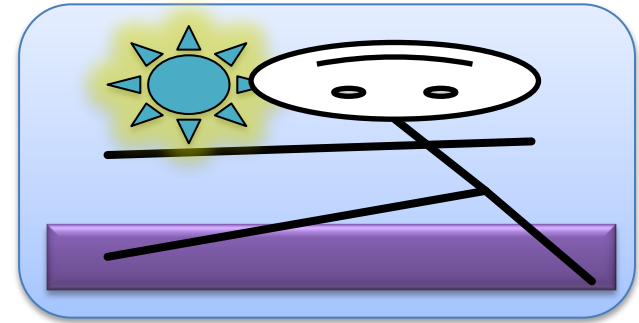
Allows the user to specify a limit on the command line after which no further transformations can be performed

# “Autochop” harness

SNC's  
max\_opts



-02

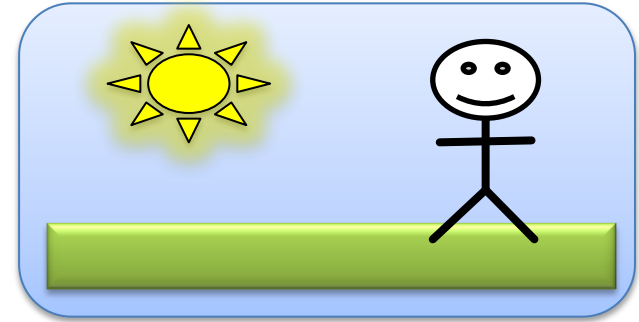


# “Autochop” harness

SNC's  
max\_opts

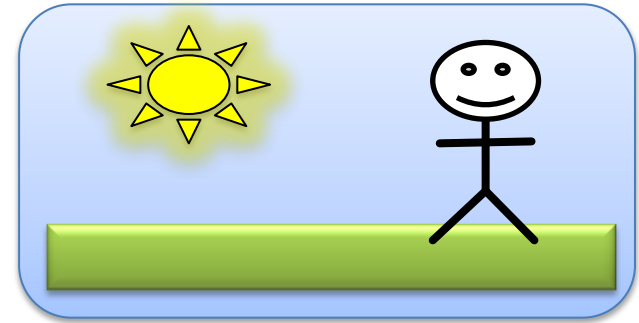
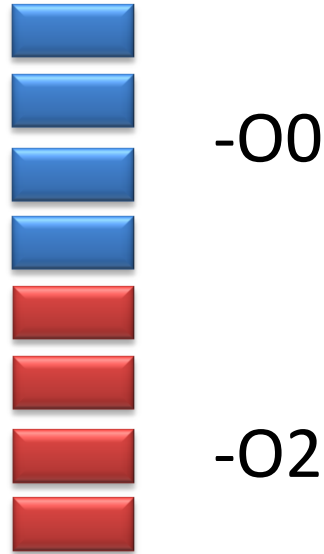


-00



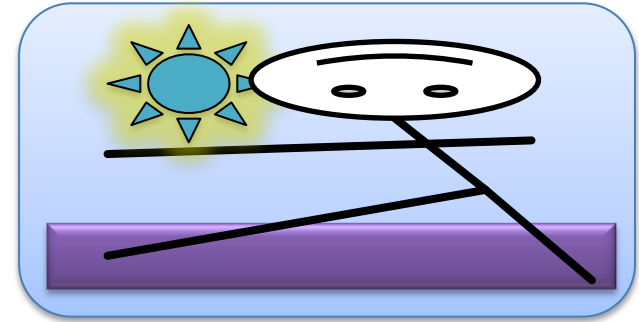
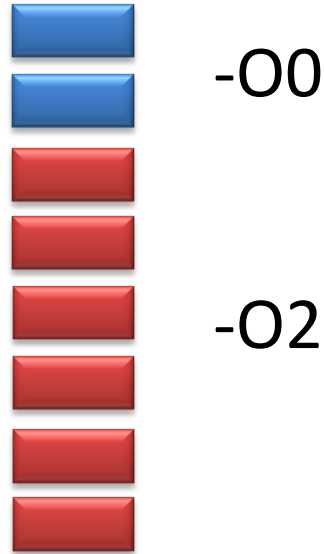
# “Autochop” harness

SNC's  
max\_opts



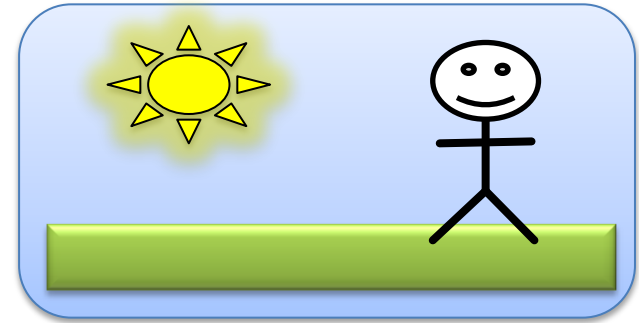
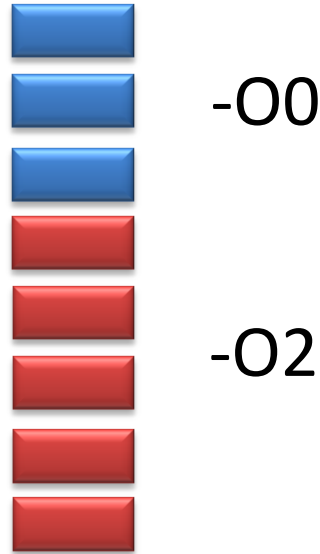
# “Autochop” harness

SNC's  
max\_opts



# “Autochop” harness

SNC's  
max\_opts



# “Autochop” harness

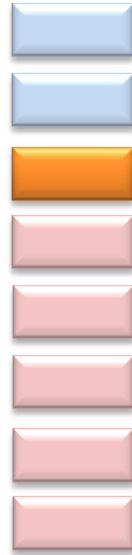
SNC's  
max\_opts



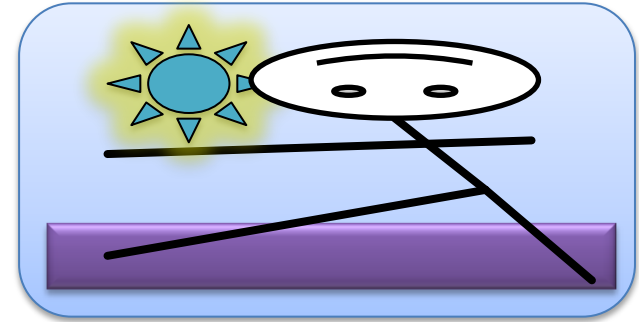
We've found the  
game source file  
with the bad  
transformation

# “Autochop” harness

SNC's  
max\_opts



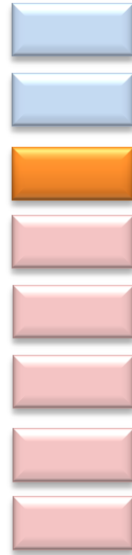
*-Xmax\_opts=2048*



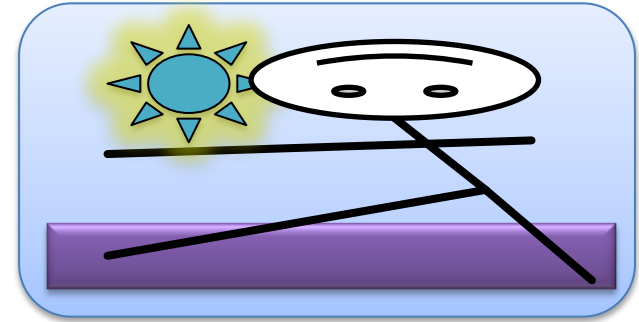


# “Autochop” harness

SNC's  
max\_opts

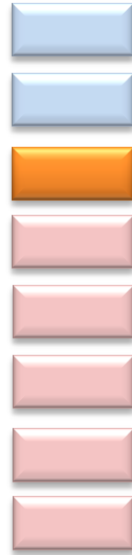


*-Xmax\_opts=1024*

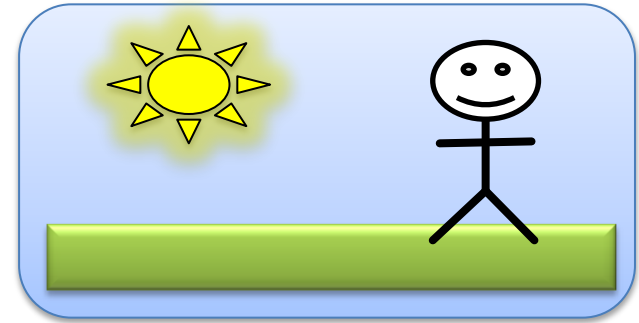


# “Autochop” harness

SNC's  
max\_opts

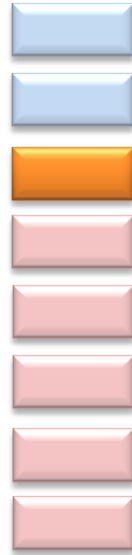


*-Xmax\_opts=512*

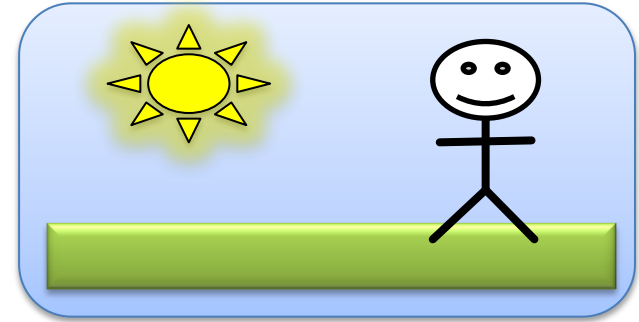


# “Autochop” harness

SNC's  
max\_opts



*-Xmax\_opts=988*

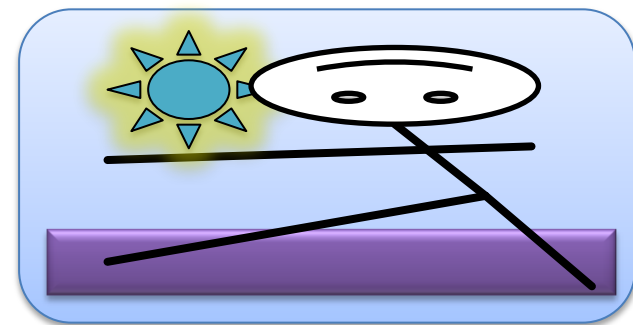


SNC's  
max\_opts

# “Autochop” harness



*-Xmax\_opts=989*



Now we've found the  
specific transformation  
causing the miscompile

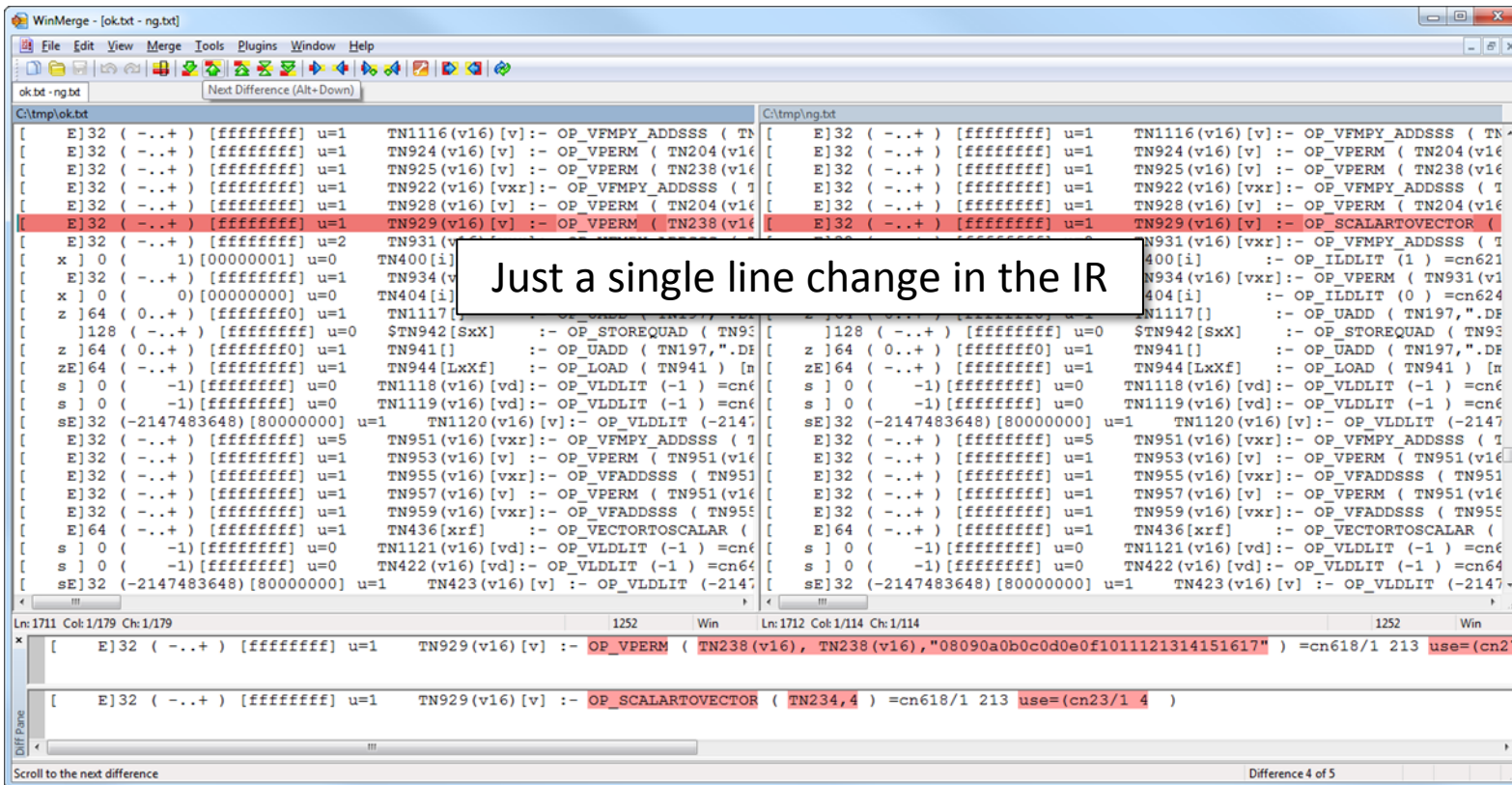
```

WinMerge - [ok.txt - ng.txt]
File Edit View Merge Tools Plugins Window Help
ok.txt - ng.txt
C:\tmp\ok.txt C:\tmp\ng.txt
opt: 966 _z14testVector3Dotv/TN387/BB1 [227:copy replaced with source
opt: 967 _z14testVector3Dotv/TN1116/BB1 [227:copy replaced with source
opt: 968 _z14testVector3Dotv/TN922/BB1 [227:copy replaced with source
opt: 969 _z14testVector3Dotv/TN931/BB1 [227:copy replaced with source
opt: 970 _z14testVector3Dotv/TN934/BB1 [227:copy replaced with source
opt: 971 _z14testVector3Dotv/TN299/BB1 [216:easy copy bypassed (24282)
opt: 972 _z14testVector3Dotv/TN299/BB1 [29:redundant instruction dele
opt: 973 _z14testVector3Dotv/TN303/BB1 [216:easy copy bypassed (24282)
opt: 974 _z14testVector3Dotv/TN303/BB1 [29:redundant instruction dele
opt: 975 _z14testVector3Dotv/TN333/BB1 [216:easy copy bypassed (24282)
opt: 976 _z14testVector3Dotv/TN333/BB1 [29:redundant instruction dele
opt: 977 _z14testVector3Dotv/TN337/BB1 [216:easy copy bypassed (24282)
opt: 978 _z14testVector3Dotv/TN337/BB1 [29:redundant instruction dele
opt: 979 _z14testVector3Dotv/TN638/BB1 [216:easy copy bypassed (24282)
opt: 980 _z14testVector3Dotv/TN638/BB1 [29:redundant instruction dele
opt: 981 _z14te
opt: 982 _z14te
opt: 983 _z14te
opt: 984 _z14te
opt: 985 _z14te
opt: 986 _z14te
opt: 987 _z14te
opt: 988 _z14te
opt: 989 _z14testVector3Dotv/TN929/BB1 [347:permute_converted_to_opsc

```

A compiler trace shows us the culprit optimization

# Narrowed down to a single difference in IR



```
C:\tmp\ok.txt | C:\tmp\ng.txt
[ E]32 ( -..+ ) [ffffff] u=1 TN1116(v16) [v] :- OP_VFMPY_ADDRSS ( TN
[ E]32 ( -..+ ) [ffffff] u=1 TN924(v16) [v] :- OP_VPERM ( TN204(v16)
[ E]32 ( -..+ ) [ffffff] u=1 TN925(v16) [v] :- OP_VPERM ( TN238(v16)
[ E]32 ( -..+ ) [ffffff] u=1 TN922(v16) [vxr] :- OP_VFMPY_ADDRSS ( T
[ E]32 ( -..+ ) [ffffff] u=1 TN928(v16) [v] :- OP_VPERM ( TN204(v16)
[ E]32 ( -..+ ) [ffffff] u=1 TN929(v16) [v] :- OP_VPERM ( TN238(v16)
[ E]32 ( -..+ ) [ffffff] u=2 TN931(v16) [vxr] :- OP_VFMPY_ADDRSS ( T
[ x ] 0 ( -..+ ) [00000001] u=0 TN400[i] :- OP_ILDLIT ( 1 ) =cn621
[ E]32 ( -..+ ) [ffffff] u=1 TN934(v16) [v] :- OP_VPERM ( TN931(v1
[ x ] 0 ( -..+ ) [00000000] u=0 TN404[i] :- OP_ILDLIT ( 0 ) =cn624
[ z ]64 ( 0..+ ) [ffffff] u=1 TN1117[] :- OP_UADD ( TN197, ".DE
[ ]128 ( -..+ ) [ffffff] u=0 $TN942[SxX] :- OP_STOREQUAD ( TN93
[ z ]64 ( 0..+ ) [ffffff] u=1 TN941[] :- OP_UADD ( TN197, ".DE
[ zE]64 ( -..+ ) [ffffff] u=1 TN944[LxXf] :- OP_LOAD ( TN941 ) [n
[ s ] 0 ( -..+ ) [ffffff] u=0 TN1118(v16) [vd] :- OP_VLDLIT ( -1 ) =cn6
[ s ] 0 ( -..+ ) [ffffff] u=0 TN1119(v16) [vd] :- OP_VLDLIT ( -1 ) =cn6
[ sE]32 (-2147483648) [80000000] u=1 TN1120(v16) [v] :- OP_VLDLIT ( -2147
[ E]32 ( -..+ ) [ffffff] u=5 TN951(v16) [vxr] :- OP_VFMPY_ADDRSS ( T
[ E]32 ( -..+ ) [ffffff] u=1 TN953(v16) [v] :- OP_VPERM ( TN951(v16)
[ E]32 ( -..+ ) [ffffff] u=1 TN955(v16) [vxr] :- OP_VFADDRSS ( TN951
[ E]32 ( -..+ ) [ffffff] u=1 TN957(v16) [v] :- OP_VPERM ( TN951(v16)
[ E]32 ( -..+ ) [ffffff] u=1 TN959(v16) [vxr] :- OP_VFADDRSS ( TN955
[ E]64 ( -..+ ) [ffffff] u=1 TN436[xf] :- OP_VECTORTOSCALAR (
[ s ] 0 ( -..+ ) [ffffff] u=0 TN1121(v16) [vd] :- OP_VLDLIT ( -1 ) =cn6
[ s ] 0 ( -..+ ) [ffffff] u=0 TN422(v16) [vd] :- OP_VLDLIT ( -1 ) =cn64
[ sE]32 (-2147483648) [80000000] u=1 TN423(v16) [v] :- OP_VLDLIT ( -2147
[ E]32 ( -..+ ) [ffffff] u=1 TN929(v16) [v] :- OP_VPERM ( TN238(v16), TN238(v16), "08090a0b0c0d0e0f1011121314151617" ) =cn618/1 213 use=(cn2
[ E]32 ( -..+ ) [ffffff] u=1 TN929(v16) [v] :- OP_SCALARTOVECTOR ( TN234, 4 ) =cn618/1 213 use=(cn23/1 4 )
```

**SNC's  
max\_opts**

**A question for the community:**

**Would this work in  
LLVM/Clang?  
(even if just at pass level)**

# ***The release process***



# Docs

**End-user documentation** is lacking

**Release notes** aimed at Clang/LLVM developers, not users

## Docs

We plan to **contribute** our documentation improvements to the **community**

# Docs

```
__mm256_round_ps  
  
SYNOPSIS  
#include <x86intrin.h>  
__m256 __mm256_round_ps(__m256 v, const int m);  
  
INSTRUCTION  
VROUNDPS  
  
DESCRIPTION  
Rounds the values stored in a packed 256-bit vector [8 x float] as specified by the byte operand. The source values are rounded to integer values and returned as floating point values.  
  
PARAMETERS  
v  
m  
A 256-bit vector of [8 x float] values.  
An immediate byte operand specifying how the rounding is to be performed.  
Bits [7:4] are reserved.  
Bit [3] is a precision exception value:  
0: A normal PE exception is used  
1: The PE field is not updated  
Bit [2] is a rounding control source:  
0: MXCSR:RC  
1: Use the RC field value  
Bit [1:0] contain the rounding control definition:  
00: Nearest  
01: Downward (toward negative infinity)  
10: Upward (toward positive infinity)  
11: Truncated  
  
RETURNS  
A 256-bit vector of [8 x float] containing the rounded values.
```

the our  
vements

# *Forward compatibility*



ABI





ABI





ABI





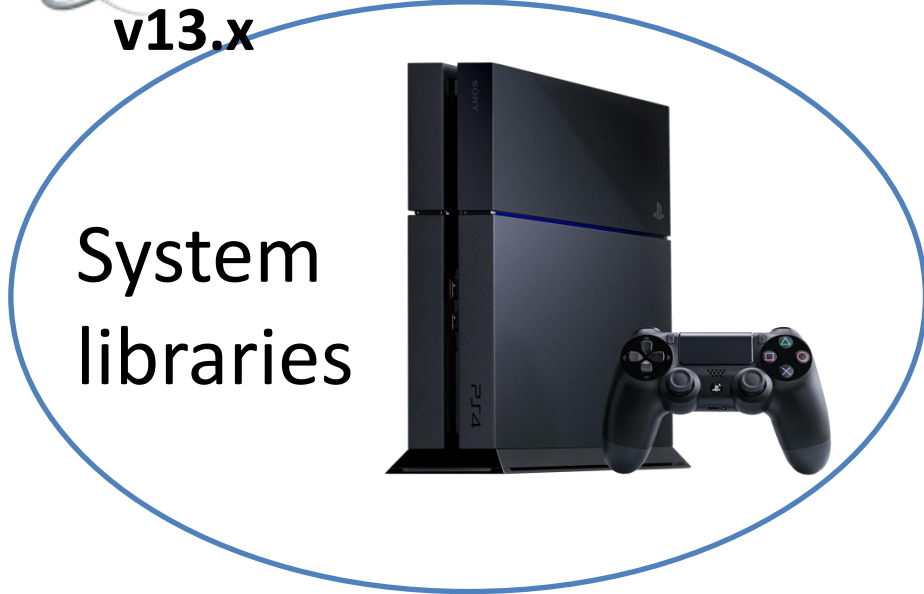
ABI







ABI



**Maintaining a stable ABI is a**

**ABI**

**MUST**

(including maintaining existing ABI bugs)

```
Administrator: C:\windows\system32\cmd.exe
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpactk_aq.c (322 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpactk_an.c (323 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpactk_ar.c (324 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpzbft_aa.c (325 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpzbft_ab.c (326 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpactk_as.c (327 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_1_00000.c (328 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpzbft_af.c (329 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpzbft_ac.c (330 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpzbft_ad.c (331 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_2_00000.c (332 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/PACKED/T_Bpzbft_ae.c (333 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00004.c (334 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00002.c (335 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00000.c (336 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00005.c (337 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00003.c (338 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00001.c (339 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00006.c (340 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00010.c (341 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00011.c (342 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00007.c (343 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00009.c (344 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00008.c (345 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00012.c (346 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00014.c (347 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00015.c (348 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00013.c (349 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00016.c (350 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00017.c (351 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00019.c (352 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_00.c (353 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_3_00018.c (354 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_09.c (355 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_17.c (356 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_15.c (357 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_31.c (358 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_33.c (359 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_63.c (360 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_32.c (361 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_16.c (362 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_64.c (363 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_01.c (364 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_08.c (365 of 366)
PASS: SN C++ IA64 ABI Tests :: struct_layout_tests/test_bitfield_07.c (366 of 366)
Testing Time: 1232.96s
Expected Passes : 350
Expected Failures : 1
Unsupported Tests : 15
$
```

We have created a full IA64 ABI test suite

# ABI TEST SUITE

We hope to **contribute** our test suite to the **community**

(some logistics still to be worked out)

# *Developer reaction*

# Developer Toolchain for



Paul T. Robinson  
Sony Computer Entertainment  
LLVM Dev Meeting, 7 Nov 2013



## Game Developers Love It!

"Toolchain is really nice, **link time is ~10 seconds, versus 2-4 minutes** on PC."

--Sammy Fatnassi, Eidos Montreal

"The quality of diagnostics is also incredible! It's **as pretentious as Google Search** when it comes to **correcting typos** for us and that's a good thing."

--Jean-François Marquis, Ubisoft



## Game Developers Love It!

Quotes from 3<sup>rd</sup>-party studios (not SCE):

"Clang for PS4™ is a **huge improvement over GCC** for PS3™. The **same codebase** (more or less) on the same hardware **went from ~25 minutes to ~1.5 minutes**. Clang's **improved warning and error messages** also pointed us to some very questionable legacy stuff."

--Steven Houchard, Gearbox

*But...*

#pragma  
optimize

Most requested feature by an  
**order of magnitude**

and already supported by all the other major compilers



This is the most common use-case:

# #pragma optimize

```
void CriticalToPerformance() {  
    ....  
}
```

```
void MaybeHasABugInIt() {  
    ....  
}
```

```
void AlsoCriticalToPerformance() {  
    ....  
}
```

Game runs too slowly at -O0, but is very hard to debug at -O2

# #pragma optimize

```
void CriticalToPerformance() {  
    .....
```

```
    #pragma optimize off  
    void MaybeHasABugInIt() {  
        .....
```

```
    }  
    #pragma optimize on
```

```
    void AlsoCriticalToPerformance() {  
        .....
```

```
    }
```

Solution: use a pragma to selectively disable optimization on a small set of functions to be debugged

We proposed this on the mailing lists, but it is a major change and we got a limited response

# #pragma optimize

```
void CriticalToPerformance() {  
    ....  
}  
  
__attribute__((optnone))  
void MaybeHasABugInIt() {  
    ....  
}  
  
void AlsoCriticalToPerformance() {  
    ....  
}
```

Short term solution: function level attribute to disable optimization

Not user-friendly for more than a very small number of functions at a time!

# #pragma optimize

```

#if SNC
    #define OPT_OFF ...
    #define OPT_ON ...
#elif GCC
    #define OPT_OFF ...
    #define OPT_ON ...
#elif MSVC
    #define OPT_OFF ...
    #define OPT_ON ...
#elif CLANG
    ;, -(
#endif

```

Many of our users abstract this away behind a compiler-independent interface. Function attribute does not fit this model!

We still need a range-based solution

# *In summary*

Our initial experience with  
Clang and LLVM has been  
**very positive**

**Thanks to all of you who  
helped make Clang and  
LLVM great!**

There are still **improvements**  
that can be made...

**We will be working alongside  
you to make them**