Subject: : AmigaOS4
Topic: : LTO in AmigaOS4 gcc
Re: LTO in AmigaOS4 gcc
Author: : kas1e
Date: : 2019/2/16 17:13:21
URL:

So i give it a go and build gcc with LTO enabled. All i need its to add to the /amiga/adtools/gcc-build/features.mk "lto" to enabled languages as well as add "--enabled-lto" option. As i read everywhere adding "lto" to enabled-languages there is not necessary, as once configure scripts will find "--enable-lto" then will add it for you, but then i anyway add it and it works. So, my current features.mk with c,c++,objc,objc++,fortran and lto looks like this:

```
 MAJOR_VERSION:=$(word 1, $(subst ., , $(VERSION)))

FEATURES=/
        --enable-languages=c,c++,objc,obj-c++,fortran,lto /
        --enable-haifa /
        --enable-lto /
        --enable-sjlj-exceptions  /
        --disable-libstdcxx-pch /
        --disable-tls

# Check, if major version is greater than or equals to 8
ifeq ($(shell test $(MAJOR_VERSION) -ge 8; echo $$?), 0)
FEATURES+=--enable-threads=amigaos
endif
```

Then i clean my previous build, run "make -C native-build gcc-cross CROSS_PREFIX=/usr/local/amiga" and in end have new gcc:

```
 $ ppc-amigaos-gcc -v
Using built-in specs.
COLLECT_GCC=ppc-amigaos-gcc
COLLECT_LTO_WRAPPER=/usr/local/amiga/libexec/gcc/ppc-amigaos/8.2.0/lto-wrapper.exe
Target: ppc-amigaos
Configured with: /amiga/adtools/gcc/repo/configure --with-bugurl=https:
//github.com/sba1/adtools/issues --with-pkgversion='adtools build 8.2.0' --target=ppc-amigaos --prefix=/usr/local/
amiga --enable-languages=c,c++,objc,obj-c++,fortran,lto --enable-haifa --enable-lto --enable-sjlj-exceptions --dis
able-libstdcxx-pch --disable-tls --enable-threads=amigaos
```

Thread model: amigaos
gcc version 8.2.0 (adtools build 8.2.0)

Then, i trying to find some benchmarks, some testcases which will show that lto is works for sure, etc, but fail. Only find some texts with descriptions how it works, with some pseudo-asm and stuff, so to only understand that LTO is another optimisation, which can do some more low-level optimisation.

So all what i test at moment, is create such a simple test case:

```
 #include <stdio.h>
#include <strings.h>

int *ptr;

void* my_malloc(size_t nbytes)
{
        if (nbytes == 0)
                return NULL;
         /* more code here */
         return ptr;
}

int main()
{

 my_malloc(100);
 my_malloc(100);
 my_malloc(100);
 my_malloc(100);

}
```

Then i build 2 versions of it:

With -O3 , but no LTO:

$ ppc-amigaos-gcc -O3 test_lto1.c -o test_lto1_no_lto

With -O3 , but with LTO:

$ ppc-amigaos-gcc -O3 -flto test_lto1.c -o test_lto1_with_lto

Now, size of the produced binary without LTO are 6176, size of binary produced with LTO enable are 6107. Difference are that version build with LTO are smaller on 69 bytes.

Then i disassmble it to asm code to see how it looks like, so, version WITHOUT lto, looks like this :

```
$ ppc-amigaos-objdump -d test_lto1_no_lto

test_lto1_no_lto:     file format elf32-amigaos


Disassembly of section .text:

01000074 <_start>:
 1000074:       94 21 ff c0     stwu    r1,-64(r1)
 1000078:       7c 08 02 a6     mflr    r0
 100007c:       3d 20 01 01     lis     r9,257
 1000080:       90 01 00 44     stw     r0,68(r1)
 1000084:       bf 21 00 24     stmw    r25,36(r1)
 1000088:       7c 79 1b 78     mr      r25,r3
 100008c:       7d bb 6b 78     mr      r27,r13
 1000090:       90 a9 10 30     stw     r5,4144(r9)
 1000094:       3d 20 01 02     lis     r9,258
 1000098:       7c bd 2b 78     mr      r29,r5
 100009c:       83 e5 02 78     lwz     r31,632(r5)
 10000a0:       39 a9 90 2c     addi    r13,r9,-28628
 10000a4:       7c 9a 23 78     mr      r26,r4
 10000a8:       80 1f 00 3c     lwz     r0,60(r31)
 10000ac:       7f e3 fb 78     mr      r3,r31
 10000b0:       7c 09 03 a6     mtctr   r0
 10000b4:       4e 80 04 21     bctrl
 10000b8:       81 5d 02 9e     lwz     r10,670(r29)
 10000bc:       3d 20 01 01     lis     r9,257
 10000c0:       3d 60 01 01     lis     r11,257
 10000c4:       3c 80 01 00     lis     r4,256
 10000c8:       93 eb 10 38     stw     r31,4152(r11)
 10000cc:       7f e3 fb 78     mr      r3,r31
 10000d0:       80 0a 00 10     lwz     r0,16(r10)
 10000d4:       38 84 10 00     addi    r4,r4,4096
 10000d8:       38 a0 00 34     li      r5,52
 10000dc:       90 09 10 44     stw     r0,4164(r9)
 10000e0:       3d 20 01 01     lis     r9,257
 10000e4:       80 1f 01 a8     lwz     r0,424(r31)
 10000e8:       91 49 10 48     stw     r10,4168(r9)
 10000ec:       7c 09 03 a6     mtctr   r0
 10000f0:       4e 80 04 21     bctrl
 10000f4:       7c 7c 1b 79     mr.     r28,r3
 10000f8:       40 82 00 4c     bne     1000144 <_start+0xd0>
 10000fc:       80 1f 02 3c     lwz     r0,572(r31)
 1000100:       3c 80 00 03     lis     r4,3
 1000104:       7f e3 fb 78     mr      r3,r31
 1000108:       60 84 80 0e     ori     r4,r4,32782
 100010c:       3b a0 00 14     li      r29,20
 1000110:       7c 09 03 a6     mtctr   r0
 1000114:       4e 80 04 21     bctrl
 1000118:       7f e3 fb 78     mr      r3,r31
```

```
100011c:	83 ff 00 40 	lwz	r31,64(r31)
1000120:	7f e9 03 a6 	mtctr	r31
1000124:	4e 80 04 21 	bctrl
1000128:	80 01 00 44 	lwz	r0,68(r1)
100012c:	7f a3 eb 78 	mr	r3,r29
1000130:	7f 6d db 78 	mr	r13,r27
1000134:	bb 21 00 24 	lmw	r25,36(r1)
1000138:	38 21 00 40 	addi	r1,r1,64
100013c:	7c 08 03 a6 	mtlr	r0
1000140:	4e 80 00 20 	blr
1000144:	80 1f 01 c0 	lwz	r0,448(r31)
1000148:	3c a0 01 00 	lis	r5,256
100014c:	7f e3 fb 78 	mr	r3,r31
1000150:	38 a5 10 10 	addi	r5,r5,4112
1000154:	7f 84 e3 78 	mr	r4,r28
1000158:	7c 09 03 a6 	mtctr	r0
100015c:	38 c0 00 01 	li	r6,1
1000160:	38 e0 00 00 	li	r7,0
1000164:	4e 80 04 21 	bctrl
1000168:	7c 7e 1b 79 	mr.	r30,r3
100016c:	41 82 00 d0 	beq	100023c <_start+0x1c8>
1000170:	3d 20 01 01 	lis	r9,257
1000174:	38 00 00 01 	li	r0,1
1000178:	81 69 10 2c 	lwz	r11,4140(r9)
100017c:	3d 20 01 01 	lis	r9,257
1000180:	3d 40 01 01 	lis	r10,257
1000184:	39 29 10 1c 	addi	r9,r9,4124
1000188:	3d 00 01 01 	lis	r8,257
100018c:	80 6b 00 00 	lwz	r3,0(r11)
1000190:	3d 60 01 01 	lis	r11,257
1000194:	3c c0 01 01 	lis	r6,257
1000198:	39 6b 10 24 	addi	r11,r11,4132
100019c:	90 01 00 08 	stw	r0,8(r1)
10001a0:	3c e0 01 01 	lis	r7,257
10001a4:	91 61 00 10 	stw	r11,16(r1)
10001a8:	3d 60 01 01 	lis	r11,257
10001ac:	7f 24 cb 78 	mr	r4,r25
10001b0:	91 21 00 0c 	stw	r9,12(r1)
10001b4:	7f 45 d3 78 	mr	r5,r26
10001b8:	38 c6 10 34 	addi	r6,r6,4148
10001bc:	93 c8 10 40 	stw	r30,4160(r8)
10001c0:	3d 00 01 00 	lis	r8,256
10001c4:	38 e7 10 3c 	addi	r7,r7,4156
10001c8:	39 08 02 78 	addi	r8,r8,632
10001cc:	80 1e 00 50 	lwz	r0,80(r30)
10001d0:	81 2a 10 50 	lwz	r9,4176(r10)
10001d4:	81 4b 10 54 	lwz	r10,4180(r11)
10001d8:	7c 09 03 a6 	mtctr	r0
10001dc:	55 4a 07 fe 	clrlwi	r10,r10,31
10001e0:	4e 80 04 21 	bctrl
10001e4:	80 1f 01 c8 	lwz	r0,456(r31)
```

```
10001e8:	7f c4 f3 78 	mr	r4,r30
10001ec:	7c 7d 1b 78 	mr	r29,r3
10001f0:	7f e3 fb 78 	mr	r3,r31
10001f4:	7c 09 03 a6 	mtctr	r0
10001f8:	4e 80 04 21 	bctrl
10001fc:	80 1f 01 ac 	lwz	r0,428(r31)
1000200:	7f 84 e3 78 	mr	r4,r28
1000204:	7f e3 fb 78 	mr	r3,r31
1000208:	7c 09 03 a6 	mtctr	r0
100020c:	4e 80 04 21 	bctrl
1000210:	7f e3 fb 78 	mr	r3,r31
1000214:	83 ff 00 40 	lwz	r31,64(r31)
1000218:	7f e9 03 a6 	mtctr	r31
100021c:	4e 80 04 21 	bctrl
1000220:	80 01 00 44 	lwz	r0,68(r1)
1000224:	7f a3 eb 78 	mr	r3,r29
1000228:	7f 6d db 78 	mr	r13,r27
100022c:	bb 21 00 24 	lmw	r25,36(r1)
1000230:	38 21 00 40 	addi	r1,r1,64
1000234:	7c 08 03 a6 	mtlr	r0
1000238:	4e 80 00 20 	blr
100023c:	80 1f 02 3c 	lwz	r0,572(r31)
1000240:	3c 80 00 03 	lis	r4,3
1000244:	7f e3 fb 78 	mr	r3,r31
1000248:	60 84 80 0e 	ori	r4,r4,32782
100024c:	3b a0 00 14 	li	r29,20
1000250:	7c 09 03 a6 	mtctr	r0
1000254:	4e 80 04 21 	bctrl
1000258:	4b ff ff a4 	b	10001fc <_start+0x188>

0100025c <my_malloc>:
100025c:	2f 83 00 00 	cmpwi	cr7,r3,0
1000260:	41 9e 00 10 	beq	cr7,1000270 <my_malloc+0x14>
1000264:	3d 20 01 01 	lis	r9,257
1000268:	80 69 10 4c 	lwz	r3,4172(r9)
100026c:	4e 80 00 20 	blr
1000270:	38 60 00 00 	li	r3,0
1000274:	4e 80 00 20 	blr

01000278 <main>:
1000278:	38 60 00 00 	li	r3,0
100027c:	4e 80 00 20 	blr
```

And version with lto enabled looks like this:

```
$ ppc-amigaos-objdump -d test_lto1_with_lto

test_lto1_with_lto:	file format elf32-amigaos
```

```
Disassembly of section .text:

01000074 <_start>:
 1000074:       94 21 ff c0     stwu    r1,-64(r1)
 1000078:       7c 08 02 a6     mflr    r0
 100007c:       3d 20 01 01     lis     r9,257
 1000080:       90 01 00 44     stw     r0,68(r1)
 1000084:       bf 21 00 24     stmw    r25,36(r1)
 1000088:       7c 79 1b 78     mr      r25,r3
 100008c:       7d bb 6b 78     mr      r27,r13
 1000090:       90 a9 10 30     stw     r5,4144(r9)
 1000094:       3d 20 01 02     lis     r9,258
 1000098:       7c bd 2b 78     mr      r29,r5
 100009c:       83 e5 02 78     lwz     r31,632(r5)
 10000a0:       39 a9 90 2c     addi    r13,r9,-28628
 10000a4:       7c 9a 23 78     mr      r26,r4
 10000a8:       80 1f 00 3c     lwz     r0,60(r31)
 10000ac:       7f e3 fb 78     mr      r3,r31
 10000b0:       7c 09 03 a6     mtctr   r0
 10000b4:       4e 80 04 21     bctrl
 10000b8:       81 5d 02 9e     lwz     r10,670(r29)
 10000bc:       3d 20 01 01     lis     r9,257
 10000c0:       3d 60 01 01     lis     r11,257
 10000c4:       3c 80 01 00     lis     r4,256
 10000c8:       93 eb 10 38     stw     r31,4152(r11)
 10000cc:       7f e3 fb 78     mr      r3,r31
 10000d0:       80 0a 00 10     lwz     r0,16(r10)
 10000d4:       38 84 10 00     addi    r4,r4,4096
 10000d8:       38 a0 00 34     li      r5,52
 10000dc:       90 09 10 44     stw     r0,4164(r9)
 10000e0:       3d 20 01 01     lis     r9,257
 10000e4:       80 1f 01 a8     lwz     r0,424(r31)
 10000e8:       91 49 10 48     stw     r10,4168(r9)
 10000ec:       7c 09 03 a6     mtctr   r0
 10000f0:       4e 80 04 21     bctrl
 10000f4:       7c 7c 1b 79     mr.     r28,r3
 10000f8:       40 82 00 4c     bne     1000144 <_start+0xd0>
 10000fc:       80 1f 02 3c     lwz     r0,572(r31)
 1000100:       3c 80 00 03     lis     r4,3
 1000104:       7f e3 fb 78     mr      r3,r31
 1000108:       60 84 80 0e     ori     r4,r4,32782
 100010c:       3b a0 00 14     li      r29,20
 1000110:       7c 09 03 a6     mtctr   r0
 1000114:       4e 80 04 21     bctrl
 1000118:       7f e3 fb 78     mr      r3,r31
 100011c:       83 ff 00 40     lwz     r31,64(r31)
 1000120:       7f e9 03 a6     mtctr   r31
 1000124:       4e 80 04 21     bctrl
 1000128:       80 01 00 44     lwz     r0,68(r1)
```

```
100012c:	7f a3 eb 78 	mr	r3,r29
1000130:	7f 6d db 78 	mr	r13,r27
1000134:	bb 21 00 24 	lmw	r25,36(r1)
1000138:	38 21 00 40 	addi	r1,r1,64
100013c:	7c 08 03 a6 	mtlr	r0
1000140:	4e 80 00 20 	blr
1000144:	80 1f 01 c0 	lwz	r0,448(r31)
1000148:	3c a0 01 00 	lis	r5,256
100014c:	7f e3 fb 78 	mr	r3,r31
1000150:	38 a5 10 10 	addi	r5,r5,4112
1000154:	7f 84 e3 78 	mr	r4,r28
1000158:	7c 09 03 a6 	mtctr	r0
100015c:	38 c0 00 01 	li	r6,1
1000160:	38 e0 00 00 	li	r7,0
1000164:	4e 80 04 21 	bctrl
1000168:	7c 7e 1b 79 	mr.	r30,r3
100016c:	41 82 00 d0 	beq	100023c <_start+0x1c8>
1000170:	3d 20 01 01 	lis	r9,257
1000174:	38 00 00 01 	li	r0,1
1000178:	81 69 10 2c 	lwz	r11,4140(r9)
100017c:	3d 20 01 01 	lis	r9,257
1000180:	3d 40 01 01 	lis	r10,257
1000184:	39 29 10 1c 	addi	r9,r9,4124
1000188:	3d 00 01 01 	lis	r8,257
100018c:	80 6b 00 00 	lwz	r3,0(r11)
1000190:	3d 60 01 01 	lis	r11,257
1000194:	3c c0 01 01 	lis	r6,257
1000198:	39 6b 10 24 	addi	r11,r11,4132
100019c:	90 01 00 08 	stw	r0,8(r1)
10001a0:	3c e0 01 01 	lis	r7,257
10001a4:	91 61 00 10 	stw	r11,16(r1)
10001a8:	3d 60 01 01 	lis	r11,257
10001ac:	7f 24 cb 78 	mr	r4,r25
10001b0:	91 21 00 0c 	stw	r9,12(r1)
10001b4:	7f 45 d3 78 	mr	r5,r26
10001b8:	38 c6 10 34 	addi	r6,r6,4148
10001bc:	93 c8 10 40 	stw	r30,4160(r8)
10001c0:	3d 00 01 00 	lis	r8,256
10001c4:	38 e7 10 3c 	addi	r7,r7,4156
10001c8:	39 08 02 5c 	addi	r8,r8,604
10001cc:	80 1e 00 50 	lwz	r0,80(r30)
10001d0:	81 2a 10 4c 	lwz	r9,4172(r10)
10001d4:	81 4b 10 50 	lwz	r10,4176(r11)
10001d8:	7c 09 03 a6 	mtctr	r0
10001dc:	55 4a 07 fe 	clrlwi	r10,r10,31
10001e0:	4e 80 04 21 	bctrl
10001e4:	80 1f 01 c8 	lwz	r0,456(r31)
10001e8:	7f c4 f3 78 	mr	r4,r30
10001ec:	7c 7d 1b 78 	mr	r29,r3
10001f0:	7f e3 fb 78 	mr	r3,r31
10001f4:	7c 09 03 a6 	mtctr	r0
```

```
10001f8:	4e 80 04 21 	bctrl
10001fc:	80 1f 01 ac 	lwz	r0,428(r31)
1000200:	7f 84 e3 78 	mr	r4,r28
1000204:	7f e3 fb 78 	mr	r3,r31
1000208:	7c 09 03 a6 	mtctr	r0
100020c:	4e 80 04 21 	bctrl
1000210:	7f e3 fb 78 	mr	r3,r31
1000214:	83 ff 00 40 	lwz	r31,64(r31)
1000218:	7f e9 03 a6 	mtctr	r31
100021c:	4e 80 04 21 	bctrl
1000220:	80 01 00 44 	lwz	r0,68(r1)
1000224:	7f a3 eb 78 	mr	r3,r29
1000228:	7f 6d db 78 	mr	r13,r27
100022c:	bb 21 00 24 	lmw	r25,36(r1)
1000230:	38 21 00 40 	addi	r1,r1,64
1000234:	7c 08 03 a6 	mtlr	r0
1000238:	4e 80 00 20 	blr
100023c:	80 1f 02 3c 	lwz	r0,572(r31)
1000240:	3c 80 00 03 	lis	r4,3
1000244:	7f e3 fb 78 	mr	r3,r31
1000248:	60 84 80 0e 	ori	r4,r4,32782
100024c:	3b a0 00 14 	li	r29,20
1000250:	7c 09 03 a6 	mtctr	r0
1000254:	4e 80 04 21 	bctrl
1000258:	4b ff ff a4 	b	10001fc <_start+0x188>

0100025c <main>:
100025c:	38 60 00 00 	li	r3,0
1000260:	4e 80 00 20 	blr
```

If you bored to compare, can say that start() about the same (some values only changes in some place). While in case with LTO i can't see my_malloc() function code at all. Is it lto optimisation somehow remove it, or whf, dunno. But what for sure, the same happens if i build the same test code from win32. I.e. same differences in disassembled code, and the same disappeared my_malloc(). Probabaly that can mean that -flto works the same for us too.

So now need to make some real lto test code, which will point out that it 100% works (or not).

Any idea of simple test case ?